Mario Köppen
Nikola Kasabov
George Coghill (Eds.)

LNCS 5507

Advances in Neuro-Information Processing

15th International Conference, ICONIP 2008
Auckland, New Zealand, November 2008
Revised Selected Papers, Part II

2 Part II

Springer

Lecture Notes in Computer Science 5507

Mario Köppen Nikola Kasabov
George Coghill (Eds.)

Advances in Neuro-Information Processing

15th International Conference, ICONIP 2008
Auckland, New Zealand, November 25-28, 2008
Revised Selected Papers, Part II

Springer

Volume Editors

Mario Köppen
Network Design and Research Center, Kyushu Institute of Technology
680-4, Kawazu, Iizuka, Fukuoka 820-8502, Japan
E-mail: mkoeppen@ieee.org

Nikola Kasabov
Auckland University of Technology
Knowledge Engineering and Discovery Research Institute (KEDRI)
School of Computing and Mathematical Sciences
350 Queen Street, Auckland 10110, New Zealand
E-mail: nkasabov@aut.ac.nz

George Coghill
Auckland University of Technology, Robotics Laboratory
Department of Electrical and Computer Engineering
38 Princes Street, Auckland 1142, New Zealand
E-mail: g.coghill@auckland.ac.nz

Library of Congress Control Number: 2009929942

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

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

ISSN 0302-9743
ISBN-10 3-642-03039-4 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-03039-0 Springer Berlin Heidelberg New York

springer.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12692576 06/3180 5 4 3 2 1 0

Preface

The two volumes contain the papers presented at the ICONIP 2008 conference of the Asia Pacific Neural Network Assembly, held in Auckland, New Zealand, November 25–28, 2008.

ICONIP 2008 attracted around 400 submissions, with approx. 260 presentations accepted, many of them invited. ICONIP 2008 covered a large scope of topics in the areas of: methods and techniques of artificial neural networks, neurocomputers, brain modeling, neuroscience, bioinformatics, pattern recognition, intelligent information systems, quantum computation, and their numerous applications in almost all areas of science, engineering, medicine, the environment, and business.

One of the features of the conference was the list of 20 plenary and invited speakers, all internationally established scientists, presenting their recent work. Among them: Professors Shun-ichi Amari, RIKEN Brain Science Institute; Shiro Usui, RIKEN Brain Science Institute, Japan; Andrzej Cichocki, RIKEN Brain Science Institute; Takeshi Yamakawa, Kyushu Institute of Technology; Kenji Doya, Okinawa Institute of Science and Technology; Youki Kadobayashi, National Institute of Information and Communications Technology, Japan; Sung-Bae Cho, Yonsei University, Korea; Alessandro Villa, University of Grenoble, France; Danilo Mandic, Imperial College, UK; Richard Duro, Universidade da Coruna, Spain, Andreas König, Technische Universität Kaiserslautern, Germany; Yaochu Jin, Honda Research Institute Europe, Germany; Bogdan Gabrys, University of Bournemouth, UK; Jun Wang, Chinese University of Hong Kong; Mike Paulin, Otago University, New Zealand; Mika Hirvensalo, University of Turku, Finland; Lei Xu, Chinese University of Hong Kong and Beijing University, China; Wlodzislaw Duch, Nicholaus Copernicus University, Poland; Gary Marcus, New York University, USA.

The organizers would also like to thank all special session organizers for their strong efforts to enrich the scope and program of this conference. The ICONIP 2008 conference covered the following special sessions: "Data Mining Methods for Cybersecurity," organized by Youki Kadobayashi, Daisuke Inoue, and Tao Ban, "Computational Models and Their Applications to Machine Learning and Pattern Recognition," organized by Kazunori Iwata and Kazushi Ikeda, "Lifelong Incremental Learning for Intelligent Systems," organized by Seiichi Ozawa, Paul Pang, Minho Lee, and Guang-Bin Huang, "Application of Intelligent Methods in Ecological Informatics," organized by Michael J. Watts and Susan P. Worner,"Pattern Recognition from Real-world Information by SVM and Other Sophisticated Techniques," organized by Ikuko Nishikawa and Kazushi Ikeda, "Dynamics of Neural Networks," organized by Zhigang Zeng and Tingwen Huang, "Recent Advances in Brain-Inspired Technologies for Robotics," organized by Kazuo Ishii and Keiichi Horio, and "Neural Information Processing in Cooperative Multi-Robot Systems," organized by Jose A. Becerra, Javier de Lope, and Ivan Villaverde.

Another feature of ICONIP 2008 was that it was preceded by the First Symposium of the International Neural Network Society (INNS) on New Directions in Neural Networks (NNN 2008), held November 25–25, 2008. This symposium was on the topic "Modeling the Brain and Neurvous systems," with two streams: Development and Learning and Computational Neurogenetic Modeling. Among the invited speakers were: A. Villa, J. Weng, G. Marcus, C. Abraham, H. Kojima, M. Tsukada, Y. Jin, L. Benuskova. The papers presented at NNN 2008 are also included in these two volumes.

ICONIP 2008 and NNN 2008 were technically co-sponsored by APNNA, INNS, the IEEE Computational Intelligence Society, the Japanese Neural Network Society (JNNS), the European Neural Network Society (ENNS), the Knowledge Engineering and Discovery Research Institute (KEDRI), Auckland University of Technology, Toyota USA, Auckland Sky City, and the School of Computing and Mathematical Sciences at the Auckland University of Technology. Our sincere thanks to the sponsors!

The ICONIP 2008 and the NNN 2008 events were hosted by the Knowledge Engineering and Discovery Research Institute (KEDRI) of the Auckland University of Technology (AUT). We would like to acknowledge the staff of KEDRI and especially the Local Organizing Chair Joyce DMello, the Web manager Peter Hwang, and the publication team comprising Stefan Schliebs, Raphael Hu and Kshitij Doble, for their effort to make this conference an exciting event.

March 2009

Nikola Kasabov
Mario Köppen
George Coghill

ICONIP 2008 Organization

ICONIP 2008 was organized by the Knowledge Engineering and Discovery Research Institute (KEDRI) of the Auckland University of Technology (AUT).

Conference Committee

General Chair	Nikola Kasabov
Program Co-chairs	Mario Köppen, George Coghill, Masumi Ishikawa
Publicity Chairs	Shiro Usui, Bill Howel, Ajit Narayanan, Suash Deb
Plenary Chairs	Takeshi Yamakawa, Andreas König, Tom Gedeon
Panels Chairs	Robert Kozma, Mario Fedrizzi, M.Tsukada, Stephen MacDonell
Tutorial Chairs	Sung-Bae Cho, Martin McGinnity, L. Benuskova
Special Session Chairs	Soo-Young Lee, Richard Duro, Shaoning Pang
Poster Session Chairs	Bernadete Ribeiro, Qun Song, Frances Joseph
Workshop Chairs	Mike Paulin, Irwin King, Kaori Yoshida, Napoleon H. Reyes
Demonstrations Chairs	Sue Worner, Russel Pears, Michael Defoin-Platel
Local Organizing Chair	Joyce Mello
Technical Support Chair	Peter Hwang

Track Chairs

Neurodynamics: Takeshi Aihara, Tamagawa University, Japan
Cognitive Neuroscience: Alessandro Villa, UJF Grenoble, France
Brain Mapping: Jagath Rajapakse, Nanyang Technological University, Singapore
Neural Network Learning Paradigms: Nik Kasabov, Auckland University of Technology, New Zealand
Kernel Methods and SVM: Bernardete Ribeiro, University of Coimbra, Portugal
Ensemble Methods for Neural Networks: Andre C.P.L.F. de Carvalho, University of Sao Paulo, Brazil
Information Algebra: Andrzej Cichocki, RIKEN, Japan
Neural Networks for Perception: Akira Iwata, Nagoya Institute of Technology, Japan
Neural Networks for Motoric Control: Minho Lee, Kyungpook National University, Korea

Neural Networks for Pattern Recognition: Paul Pang, Auckland University of Technology, New Zealand
Neural Networks for Robotics: Richard Duro, University of A Coruña, Spain
Neuromorphic Hardware: Leslie S. Smith, University of Stirling, UK
Embedded Neural Networks: Andreas Koenig, University of Kaiserslautern, Germany
Neural Network-Based Semantic Web, Data Mining and Knowledge Discovery:
Irwin King, The Chinese University of Hong Kong, Hong Kong
Computational Intelligence: Wlodzislaw Duch, Nicolaus Copernicus University, Poland
Bioinformatics: Sung-Bae Cho, Yonsei University, Korea
Neural Paradigms for Real-World Networks: Tom Gedeon, The Australian National University, Australia
Quantum Neural Networks: Mika Hirvensalo, University of Turku, Finland
Neural Network Implementation in Hardware and Software: George Coghill, Auckland University of Technology, New Zealand
Biologically Inspired Neural Networks: Nik Kasabov, Auckland University of Technology, New Zealand

International Technical Committee

Abbass, Hussein
Abe, Shigeo
Aihara, Takeshi
Alippi, Cesare
Ando, Ruo
Andras, Peter
Asoh, Hideki
Ban, Tao
Bapi, Raju
Barczak, Andre Luis Chautard
Barros, Allan Kardec
Becerra Permuy, José Antonio
Behera, Laxmidhar
Behrman, Elizabeth
Beliczynski, Bartlomiej
Carvalho, Andre C.P.L.F. de
Chang, Jyh-Yeong
Cho, Sung-Bae
Choi, Seungjin
Chung, I-Fang
Cichocki, Andrzej
Coghill, George
Cohen, Avis
Dauwels, Justin
de Lope, Javier
de Souto, Marcilio
Dorronsoro, Jose
Dourado, Antonio
Duch, Wlodzislaw
Duro, Richard
Elizondo, David
Erdi, Peter
Fukumura, Naohiro
Fung, Wai-keung
Furukawa, Tetsuo
fyfe, colin
Garcez, Artur
Gedeon, Tom
Grana, Manuel
Gruen, Sonja
Guo, Shanqing
Hagiwara, Katusyuki
Hammer, Barbara
Hartono, Pitoyo
Hayashi, Akira
Hayashi, Hatsuo
Hikawa, Hiroomi
Hirvensalo, Mika
Honkela, Antti
Horio, Keiichi
Huang, Kaizhu
Ikeda, Kazushi
Inoue, Daisuke
Ishida, Fumihiko
Iwata, Kazunori
Iwata, Akira
Kadone, Hideki
Kanoh, Shin'ichiro
Kasabov, Nikola
Kim, Kyung-Joong
Kimura, Shuhei
King, Irwin
Kitajima, Tatsuo
Koenig, Andreas
Koeppen, Mario
Kondo, Toshiyuki
Kurita, Takio
Kurogi, Shuichi
Lai, Weng Kin

Lee, Minho
Lendasse, Amaury
Lim, CP
Liu, Ju
Liu, Shih-Chii
Lu, Bao-Liang
Ludermir, Teresa
Mandziuk, Jacek
Matsui, Nobuyuki
Mayr, Christian
McKay, Bob
Meier, Karlheinz
Mimura, Kazushi
Miyamoto, Hiroyuki
Molter, Colin
Morie, Takashi
Morita, Kenji
Nakajima, Koji
Nakauchi, Shigeki
Nguyen-Tuong, Duy
Nishii, Jun
Nishikawa, Ikuko
Ohnishi, Noboru
Omori, Toshiaki
Ozawa, Seiichi
Pang, Paul
Patel, Leena
Peters, Jan
Phillips, Steven
Rajapakse, Jagath
Reyes, Napoleon
Ribeiro, Bernardete
Rueckert, Ulrich
Sakai, Ko
Sato, Shigeo
Sato, Naoyuki
Schemmel, Johannes
Setiono, Rudy
Shibata, Tomohiro
Shimazaki, Hideaki
Shouno, Hayaru
Silva, Catarina
Small, Michael
Smith, Leslie S.
Spaanenburg, Lambert
Stafylopatis, Andreas
Suematsu, Nobuo
Suh, Il Hong
Sum, John
Suykens, Johan
Takenouchi, Takashi
Tambouratzis, Tatiana
Tanaka, Yoshiyuki
Tang, Ke
Tateno, Katsumi
van Schaik, Andre
Villa, Alessandro
Villaverde, Ivan
Wada, Yasuhiro
Wagatsuma, Hiroaki
Watanabe, Keigo
Watanabe, Kazuho
Watts, Michael
Wu, Jianhua
Xiao, Qinghan
Yamaguchi, Nobuhiko
Yamauchi, Koichiro
Yi, Zhang
Yoshimoto, Junichiro
Zhang, Zonghua
Zhang, Liming
Zhang, Liqing
Zhang, Byoung-Tak

Additional Referees

Pong Meau Yeong, Hua Nong Ting, Sim Kok Swee, Yap Keem Siah, Shahrel Azmin Suandi, Tomas Henrique Bode Maul, Nor Ashidi Mat Isa, Haidi Ibrahim, Tan Shing Chiang, Dhanesh Ramachand Ram, Mohd Fadzli Mohd Salleh, Khoo Bee Ee

Sponsoring Institutions

Asia Pacific Neural Network Assembly (APNNA)
International Neural Network Society (INNS)
IEEE Computational Intelligence Society
Japanese Neural Network Society (JNNS)
European Neural Network Society (ENNS)
Knowledge Engineering and Discovery Research Institute (KEDRI)
Auckland University of Technology (AUT)
Toyota USA
Auckland Sky City
School of Computing and Mathematical Sciences at the Auckland University of Technology

INNS NNN 2008 Organization

INNS NNN 2008 was organized by the Knowledge Engineering and Discovery Research Institute (KEDRI) of the Auckland University of Technology (AUT).

Conference Committee

General Chair	Juyang Weng
Program Chair	Nikola Kasabov
Program Co-chairs	Mario Koeppen, John Weng, Lubica Benuskova
Local Organizing Chair	Joyce DMello
Technical Support Chair	Peter Hwang
Publishing Committee	Stefan Schliebs, Kshitij Dhoble, Raphael Hu
Symposium 1 Co-chairs	Juyang Weng, Jeffrey L. Krichmar, Hiroaki Wagatsuma
Symposium 2 Co-chairs	Lubica Benuskova, Alessandro E.P. Villa, Nikola Kasabov

Program Committee

Alessandro E.P. Villa
Anil Seth
wCharles Unsworth
Chris Trengove
Cliff Abraham
Danil Prokhorov
Frederic Kaplan
Hiroaki Wagatsuma
Igor Farkas
James Wright
Jason Fleischer
Jeffrey L. Krichmar
Jiri Sima
Juyang Weng
Kazuhiko Kawamura
Lubica Benuskova
Michael Defoin Platel
Michal Cernansky
Ming Xie
Peter Jedlicka
Rick Granger
Rolf Pfeifer
Roman Rosipal
Xiangyang Xue
Zhengyou Zhang

Reviewers

Danil Prokhorov
Ming Xie
Frederic Kaplan
Hiroaki Wagatsuma
Kaz Kawamura
Juyang Weng
Lubica Benuskova
Igor Farkas

Chris Trengove
Charles Unsworth
Roman Rosipal
Jiri Sima
Alessandro Villa
Peter Jedlicka
Michal Cernansky
Michael Defoin-Platel
Xiangyang Xue
Wickliffe Abraham
James Wright
Zhengyou Zhang
Anil Seth
Jason Fleischer
Nikola Kasabov

Sponsoring Institutions

Auckland University of Technology (AUT)
Asia Pacific Neural Network Assembly (APPNA)
International Neural Network Society (INNS)
Knowledge Engineering and Discovery Research Institute (KEDRI)
IEEE Computational Intelligence Society

Table of Contents – Part II

I Neural Network Based Semantic Web, Data Mining and Knowledge Discovery

II Neural Networks Learning Paradigm

III Kernel Methods and SVM

IV Neural Networks as a Soft Computing Technology

V Neural Networks and Pattern Recognition

VI Neuromorphic Hardware and Embedded Neural Networks

VII Machine Learning and Information Algebra

VIII Brain-Computer Interface

IX Neural Network Implementations

Table of Contents – Part I

I INNS Symposium "New Directions in Neural Networks"

II Neurodynamics

III Cognitive Neuroscience

IV Bioinformatics

V Special Session: Data Mining Methods for Cybersecurity

VI Special Session: Computational Models and Their Applications in Machine Learning and Pattern Recognition

VII Special Session: Recent Advances in Brain-Inspired Technologies for Robotics

VIII Special Session: Lifelong Incremental Learning for Intelligent Systems

IX Special Session: Dynamics of Neural Networks

X Special Session: Applications of Intelligent Methods in Ecological Informatics

XI Special Session: Pattern Recognition from Real-World Information by SVM and Other Sophisticated Techniques

XII Special Session: Neural Information Processing in Cooperative Multi-robot Systems

XIII WORKSHOP: Hybrid and Adaptive Systems for Real-Time Robotics Vision and Control

XIV WORKSHOP: Neurocomputing and Evolving Intelligence – NCEI 2008

Part I

Neural Network Based Semantic Web, Data Mining and Knowledge Discovery

A Novel Method for Manifold Construction

Wei-Chen Cheng and Cheng-Yuan Liou*

Department of Computer Science and Information Engineering
National Taiwan University
Republic of China
cyliou@csie.ntu.edu.tw

Abstract. This paper presents a distance invariance method to construct the low dimension manifold that preserves the neighborhood topological relations among data patterns. This manifold can display close relationships among patterns.

1 Introduction

Dimension reduction in the manifold space can configure topological relationships among patterns. Foundations for various data manifolds have been set down for the factorial components [8], oblique transformation [11], ICA [19], generalized adaline [21]. They have been successfully applied in various temporal data analyses [20]. The principal component analysis (PCA) and multidimensional scaling (MDS) [18] are well established linear models that have been developed for such reduction. Many nonlinear reduction algorithms [2][6][9] have been designed with varying degrees of success. The Isomap [1][17] extends MDS by using the geodesic distance to construct the nonlinear manifold. The Locally Linear Embedding (LLE) [14] computes certain linear model coefficients to maintain the local geometric properties in the manifold. Both Isomap and LLE have distinguished performances. Isomap has been applied to find intrinsic curvature manifold, such as a fishbowl surface. The local distance preservation (LDP) method has been proposed to learn the latent relations of data [12] directly. This paper further devises a distance invariance method based on the LDP and presents new applications.

2 Method

Suppose there are P patterns distributed in a D-dimensional pattern space, $X = \{\mathbf{x}^j,\ j = 1, ..., P\}$. Each pattern point, $\mathbf{x}^j$, is a D-dimensional column vector and has a corresponding mapped cell, $\mathbf{y}^j$, in the output manifold space. The positions of the cells in the output space are $Y = \{\mathbf{y}^j,\ j = 1, ..., P\}$. Each output, $\mathbf{y}^j$, is a M-dimensional column vector. In the pattern space, for a pattern point $\mathbf{x}^p$, the set of those points whose distances to $\mathbf{x}^p$ are less than r are included

* Corresponding author.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 3–10, 2009.

in the set $U(p,r)$, where r denotes the radius of neighborhood region in the pattern space. The notation $|U(p,r)|$ denotes the number of points in the set $U(p,r)$. Rewrite the SIR energy function [10][12], $E = \pm\frac{1}{2}\sum\limits_p\sum\limits_q [d(\mathbf{y}^p, \mathbf{y}^q)]^2$, get

$$E(r) = \sum_p E^p(r) = \frac{1}{4}\sum_p \sum_{\mathbf{x}^q \in U(p,r)} \left(\|\mathbf{y}^p - \mathbf{y}^q\|^2 - \|\mathbf{x}^p - \mathbf{x}^q\|^2\right)^2, \tag{1}$$

where

$$E^p(r) = \sum_{\mathbf{x}^q \in U(p,r)} \left(\|\mathbf{y}^p - \mathbf{y}^q\|^2 - \|\mathbf{x}^p - \mathbf{x}^q\|^2\right)^2.$$

This is a distance invariance energy function for the construction of manifold. Note that there is no percise energy function for SOM, see the preface in [6]. The difference between (1) and MDS [18] is that (1) is a function of r. All pattern points are used in (1) when r is very large. Few neighbors are used when r is small. The LDP algorithm that adjusts the cell position, $\mathbf{y}^j$, in the output manifold space is as follows.

LDP Algorithm

```
1. Initialize the output set $Y$.
2. Assign a value to $r$.
3. For each epoch
4.     For every pair patterns $\mathbf{x}^p$ and $\mathbf{x}^q$, adjust their cell positions by
```

$$\mathbf{y}^p \longleftarrow \mathbf{y}^p - \eta\frac{\partial E(r)}{\partial \mathbf{y}^p},\ \mathbf{y}^q \longleftarrow \mathbf{y}^q - \eta\frac{\partial E(r)}{\partial \mathbf{y}^q}. \tag{2}$$

```
5.     Reduce $r$.
6. End For
```

In the above algorithm, η is a learning rate. The gradient is

$$\frac{\partial E(r)}{\partial \mathbf{y}^p} = 2 \times \sum_{\mathbf{x}^q \in U(p,r)} \left(\|\mathbf{y}^p - \mathbf{y}^q\|^2 - \|\mathbf{x}^p - \mathbf{x}^q\|^2\right)(\mathbf{y}^p - \mathbf{y}^q),$$

and the coefficient 2 can be absorbed into the learning rate. The computational complexity for calculating the pairwise distance is $O\left(DP^2\right)$ and finding the neighbors for every pattern is $O\left(P^2\right)$. To explain the idea of this algorithm, one may imagine that there are P labeled balls (cells) on a flat table. This table resembles the low dimensional space, M. These balls are free to move on the table. So, they are constrained on this M-dimensional space. Each ball is labeled with its corresponding pattern p. The position of the pth ball on the table is $\mathbf{y}^p$. The algorithm seeks a ball distribution Y in M that can reveal and resemble the neighborhood topology of X in D. The system energy (1) exerts a force to the current distribution Y to force it maximally similar to the distribution X. Of course, one can construct a Gibbs type system for the energy (1) to relax the table distribution as a whole and solve the distribution Y. Here, we prefer a sequential operation, Step 4, by using the force to redistribution the balls.

There are many flexible ways to assign the initial cell positions in Step 1. For example, one may apply the linear projection methods, such as PCA and MDS, to project all pattern points onto the M-dimensional space. Other developed methods, LLE and Isomap, can also be used for the initial assignment.

In Step 2, r denotes the neighborhood radius of the hypersphere in the pattern space. Those points inside the hypersphere are used in the energy (1). During the end of the training process, the value of r is shrunk to the minimum distance among all pattern pairs. We set the initial value r to be the maximum distance among all pattern pairs and reduce r linearly or exponentially. Note that when r is kept at a large constant for a long period, the evolution of cell positions may not stop and not reach convergence. The algorithm stops the evolution by fading out those far neighbors manually. The computational complexity of updations is

$$\text{(number of iterations)} \times O\left(MP \max_{\mathbf{x}^p} |U(p,r)|\right). \tag{3}$$

Note that the LDP organizes the positions of the output cells based on the distance relations instead of the pattern vectors used in SOM [6]. These vectors need absolute coordinates. The relative distances are readily available in many applications. The positions of the cells will not be fixed in regular positions as those in SOM. There is no synaptic weight attached to each cell that indicates its absolute constellation in the pattern space.

3 Experiments on Artificial Data

3.1 Swiss Roll Dataset

In this example, we show that the sampling density affects the LLE manifold. The Swiss roll equation is

$$\left(\sqrt{\frac{u}{2\pi}}\sin(3\pi u), \sqrt{\frac{u}{2\pi}}\cos(3\pi u), v\right), 0 \le u \le 1, -\frac{3}{10} \le v \le \frac{3}{10}. \tag{4}$$

We uniformly sample data points along the variable v in the range $\left[-\frac{3}{10}, \frac{3}{10}\right]$ and non-uniformly along u. Let $\mathbf{r}(u)$ be the equation of the curve,

$$\mathbf{r}(u) = \left(\sqrt{\frac{u}{2\pi}}\sin(3\pi u), \sqrt{\frac{u}{2\pi}}\cos(3\pi u)\right), 0 \le u \le 1. \tag{5}$$

The arc length of $\mathbf{r}(u)$ is

$$f(u) = \int_0^u \sqrt{\frac{\partial \mathbf{r}(u')}{\partial u'} \cdot \frac{\partial \mathbf{r}(u')}{\partial u'}} du', \tag{6}$$

where $f(u)$ is the arc length with respect to u. We sample along $f(u)$ and use f^{-1} to compute the u value. With this u and v, we can find a corresponding

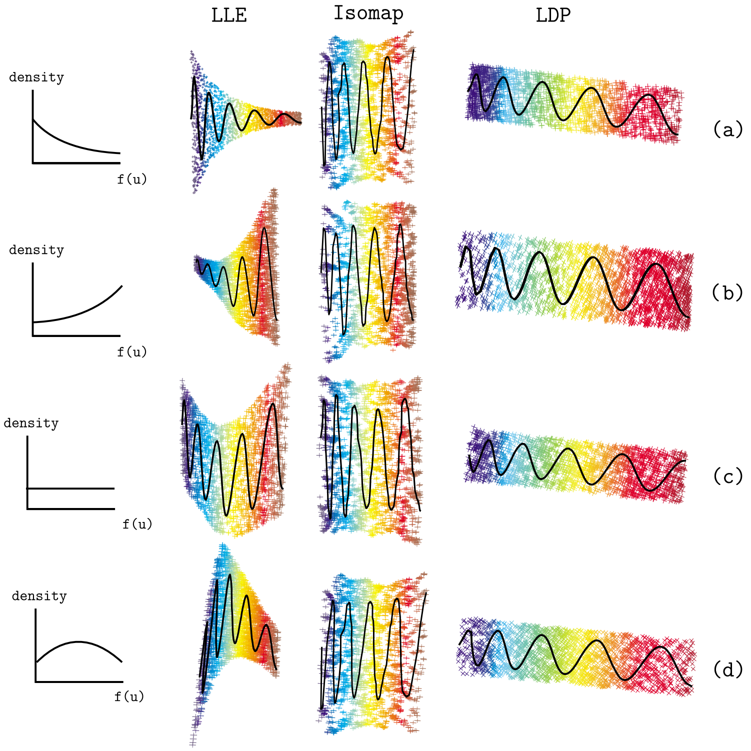


Fig. 1. The manifolds of LLE, Isomap and LDP with different sampling densities along the arc length $f(u)$ plotted in the left diagrams

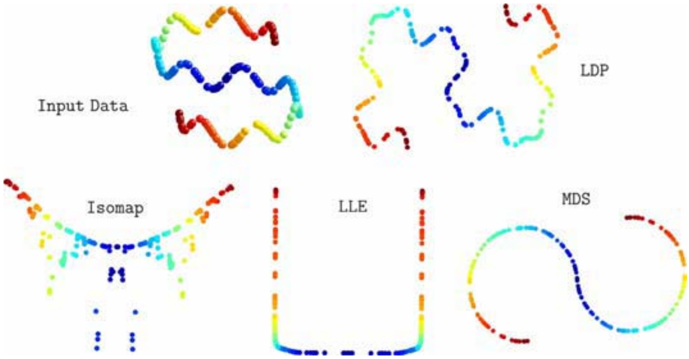


Fig. 2. The left-top image is the curve which has two kinds of structures. This curve has a S-shape on the xy-plane and a sinusoid curve along the z-axis. The manifolds obtained by using LDP, Isomap, LLE and MDS are plotted with their labels.

point on the surface by using the roll formula (4). Using this sampling technique, the probability density of a point may be uniform or non-uniform along the arc length. We can design any sampling densities along the variables, u and v. Figure 1 shows the manifolds obtained by different algorithms using different density distributions. The diagrams on the left column plot the density distributions with respect to $f\,(u)$. In this column, the horizontal coordinate, x-axis, is $f\,(u)$, and the vertical coordinate, y-axis, is the density. Figure 1(a) shows the manifolds of unbalanced sampling. The blue area of the roll has dense sampling points in unit area and the red area has low density of points. Figure 1(b) shows the manifolds that the red part of the roll has high density. Figure 1(c) shows the manifolds that the density is even along the arc length. Figure 1(d) shows the manifolds that the yellow area (middle portion) has high density. From Figure 1, we see that Isomap and LDP are not affected much by the density distributions. This is because every pattern has its correspondent cell in the output space. The density of each pattern is equal to the density of its cell. There is no probability manipulation [9]. There is no clustering and no LVQ in LDP, see the preface in Kohonen's book [6]. The LDP is developed only for data visualization on the manifold space. The LDP transfers the coordinates of all patterns to the manifold space. The sinusoid curve in the LLE manifold will fluctuate with the density.

3.2 S-Curve

The LDP fully utilizes the output space to maintain both global and local structures. As an example, the pattern points in Figure 2 have two kinds of structures. These points globally form a S-curve on the xy-plane and locally form a sinusoid curve along the z-axis. The points which have the same color are from the same input data. The LDP reveals these two kinds of information. Both Isomap and LLE with parameter, $K = 12$, show incorrect structures. MDS maps data onto a projection plane and reveals the S-curve structure only.

4 Experiments on Real Data

4.1 Phylogenetic Tree

The phylogenetic tree can be used to visualize the inter-relations among species. Given a set of distance relations, we plan to construct the edge lengths of a given tree that meets the distance relations with its path lengths. Rewrite the SIR energy function [10][12], $E = \pm\frac{1}{2}\sum\limits_{p}\sum\limits_{q}[d(\mathbf{y}^p, \mathbf{y}^q)]^2$, for the path lengths and get

$$\hat{E} = \sum_{p}\sum_{q}\left(t\,(p,q) - \|\mathbf{x}^p - \mathbf{x}^q\|\right)^2, \tag{7}$$

where $t\,(p,q)$ denotes the path length from the leaf node p to the leaf node q. The tree is an undirected binary tree. Its edges have no direction. The cells are the leaf nodes. The tree has P leaf nodes and $2P-2$ edges. Sattath [15] suggested using the least square method with non-negative constraint to modify the edge lengths of the tree. Assume that the lengths of the $2P-2$ edges are variables, $\mathbf{z} = \left[z_1, \ldots, z_{(2P-2)}\right]^T$. We construct a matrix for a known tree, $A_{\left(\frac{P(P-1)}{2}\right) \times (2P-2)}$. In A, the row indices denote the tree paths corresponding to input pairs, $\{\left(\mathbf{x}^i, \mathbf{x}^j\right),$ $i = 1, ..., P;\ j = i+1, ..., P\}$, and the column indices denote the $2P-2$ edges of the tree. The element, a_{ij}, of the matrix A is set as

$$a_{ij} = \begin{cases} 0 \text{ , when the path } i \text{ doesn't contain the edge } j. \\ 1 \text{ , when the path } i \text{ includes the edge } j. \end{cases} . \tag{8}$$

Let $\delta = \left[\left\|\mathbf{x}^1 - \mathbf{x}^2\right\|, \left\|\mathbf{x}^1 - \mathbf{x}^3\right\|, \ldots, \left\|\mathbf{x}^{P-1} - \mathbf{x}^P\right\|\right]^T$ be a $\frac{P(P-1)}{2}$ -by-1 column vector that consists of all distances between data pairs. We require that the edge lengths $\mathbf{z}$ satisfy

$$A\mathbf{z} \approx \delta, \text{ subject to } \mathbf{z} \geq \mathbf{0}. \tag{9}$$

Employ the idea of LDP and write

$$\sum_{j=1}^{2P-2} a_{ij} z_j \approx \delta_i, \text{ if } \delta_i \leq r \tag{10}$$

$$\text{subject to } \mathbf{z} \geq \mathbf{0}. \tag{11}$$

We adjust the edge lengths $\mathbf{z}$ by the gradient descent method and restrict the values of $\mathbf{z}$ to be larger than or equal to zero. We expect that the equation (10) can minimize the energy function, $\hat{E}(r)$, after convergence

$$\hat{E}(r) = \sum_p \sum_{\mathbf{x}^q \in U(p,r)} \left(t(p,q) - \left\|\mathbf{x}^p - \mathbf{x}^q\right\|\right)^2. \tag{12}$$

The value of r is reduced during the training.

The set of data used in the experiment is Case's data [3] that contains the immunological distances among nine frog (Rana) species. In simulations, we employ the technique UPGMA (Unweighted Pair Group Method with Arithmetic mean) [16] to build a tree and use LDP to estimate the edge lengths of the tree.

Figure 4 shows the estimated tree lengths by LDP. The lengths by Sattath [15] is also plotted in this figure for comparison. The LDP obtains very different subtree for the five species, R. aurora, R. boylii, R. cascadae, R. muscosa, and R. pretiosa. After convergence, we compute the performance using the formula, $MDI(r)$, for the LDP and Sattath's method. The measurement of distance invariance, $MDI(r)$, is

$$MDI(r) = \frac{1}{\sum_p |U(p,r)|} \sum_p \sum_{\mathbf{x}^q \in U(p,r)} \frac{\sqrt{\left(t(p,q) - \left\|\mathbf{x}^p - \mathbf{x}^q\right\|\right)^2}}{\left\|\mathbf{x}^p - \mathbf{x}^q\right\|}. \tag{13}$$

The performance is plotted in Figure 3. In this figure, the x-axis denotes r and y-axis denotes the corresponding $MDI(r)$ in (13). The performance shows that LDP obtains very precise length information for those close distance species.

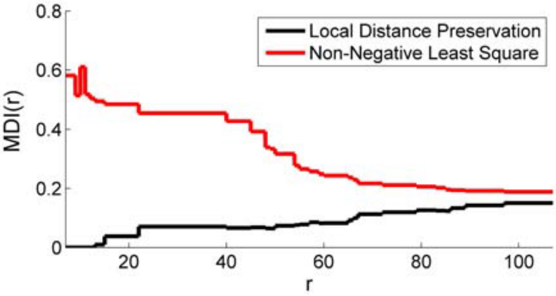


Fig. 3. The performance comaprison, $MDI\left(r\right)$

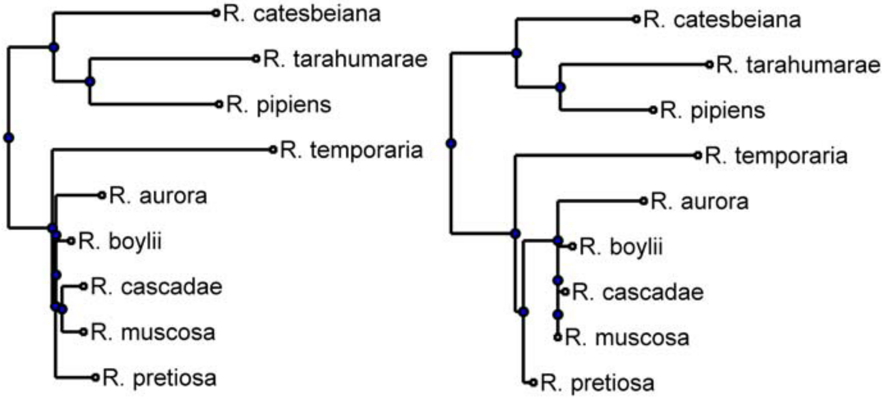


Fig. 4. The trees are constructed by UPGMA . The edge lengths are obtained by LDP (left) and Sattath's method (right).

4.2 Summary

Distortion analysis [13][7] has been introduced to study the formation mechanism of the SOM [5]. The lack of precise energy function [4] of the SOM. This paper devised a distance invariance energy for the manifold construction. It is a precise energy function. The manifold is stable for patterns with different densities. It can resolve detailed substructures. There is no LVQ, no clustering and no probability manipulation in the method, see the preface in [6]. The method is developed for data visualization only. The flat output space Y may be set to a curved Reimann space.

The manifold can be used in many applications, such as time series, chain, and tree length. It is relatively difficult to display chains or tree links in SOM. The LLE manifold is sensitive to the density distribution of patterns. The LDP and Isomap manifolds are much stable when patterns have different density distributions. The computational complexity of Isomap is $O\left(P^3\right)$. It is higher than that of LLE. The complexity of LDP is $O\left(MP^2\right)$. M is very small in the dimension reduction.

Acknowledgement. This work is supported by National Science Council.

References

1. Balasubramanian, M., Schwartz, E.L.: The Isomap Algorithm and Topological Stability. Science 295, 7 (2002)
2. Bishop, C.M., Svensen, M., Williams, C.K.I.: GTM: The Generative Topographic Mapping. NCRG/96/015 (1997)
3. Case, S.M.: Biochemical Systematics of Members of the Genus Rana Native to Western North America. Systematic Zoology 27, 299–311 (1978)
4. Erwin, E., Obermayer, K., Schulten, K.: Self-Organizing Maps: Ordering, Convergence Properties and Energy Functions. Biological Cybernetics 67, 47–55 (1992)
5. Kohonen, T.: Self-Organized Formation of Topologically Correct Feature Maps. Biological Cybernetics 43, 59–69 (1982)
6. Kohonen, T.: Self-Organization and Associative Memory, 2nd edn., pp. 119–157. Springer, Berlin (1988)
7. Kohonen, T.: Comparison of SOM Point Densities Based on Different Criteria. Neural Computation 11, 2081–2095 (1999)
8. Liou, C.-Y., Musicus, B.R.: Separable Cross-Entropy Approach to Power Spectrum Estimation. IEEE Transactions on Acoustics, Speech and Signal Processing 38, 105–113 (1990)
9. Liou, C.-Y., Tai, W.-P.: Conformality in the Self-Organization Network. Artificial Intelligence 116, 265–286 (2000)
10. Liou, C.-Y., Chen, H.-T., Huang, J.-C.: Separation of Internal Representations of the Hidden Layer. In: Proceedings of the International Computer Symposium, Workshop on Artificial Intelligence, pp. 26–34 (2000)
11. Liou, C.-Y., Musicus, B.R.: Cross Entropy Approximation of Structured Gaussian Covariance Matrices. IEEE Transaction on Signal Processing 56, 3362–3367 (2006)
12. Liou, C.-Y., Cheng, W.-C.: Manifold Construction by Local Neighborhood Preservation. In: Ishikawa, M., Doya, K., Miyamoto, H., Yamakawa, T. (eds.) ICONIP 2007, Part II. LNCS, vol. 4985, pp. 683–692. Springer, Heidelberg (2008)
13. Luttrell, S.: Code Vector Density in Topographic Mappings: Scalar Case. IEEE Transactions on Neural Networks 2, 427–436 (1991)
14. Roweis, S.T., Saul, L.K.: Nonlinear Dimensionality Reduction by Locally Linear Embedding. Science 290, 2323–2326 (2000)
15. Sattath, S., Tversky, A.: Additive Similarity Trees. Psychometrika 42, 319–345 (1977)
16. Sokal, R.R., Sneath, P.H.A.: Principles of Numerical Taxonomy. W. H. Freeman, San Francisco (1963)
17. Tenenbaum, J., Silva, V., Langford, J.C.: A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science 290, 2319–2323 (2000)
18. Torgerson, W.S.: Multidimensional Scaling, I: Theory and Method. Psychometrika 17, 401–419 (1952)
19. Wu, J.-M., Chiu, S.-J.: Independent Component Analysis Using Potts Models. IEEE Transactions on Neural Networks 12, 202–212 (2001)
20. Wu, J.-M., Lu, C.-Y., Liou, C.-Y.: Independent Component Analysis of Correlated Neuronal Responses in Area MT. In: Proceedings of the International Conference on Neural Information Processing, pp. 639–642 (2005)
21. Wu, J.-M., Lin, Z.-H., Hsu, P.-H.: Function Approximation Using Generalized Adalines. IEEE Transactions on Neural Networks 17, 541–558 (2006)

A Non-linear Classifier for Symbolic Interval Data Based on a Region Oriented Approach

Renata M.C.R. de Souza and Diogo R.S. Salazar

Centro de Informatica - CIn / UFPE, Av. Prof. Luiz Freire, s/n,
Cidade Universitaria CEP: 50740-540 - Recife - PE - Brasil
{rmcrs,drss2}@cin.ufpe.br

Abstract. This paper presents a non-linear classifier based on a region oriented approach for interval data. Each example of the learning set is described by a interval feature vector. Concerning the learning step, each class is described by a region (or a set of regions) in $\Re^p$ defined by hypercube of the objects belonging to this class. In the allocation step, the assignment of a new object to a class is based on a suitable L_r Minkowski distance between intervals. Experiments with two synthetic interval data sets have been performed in order to show the usefulness of this classifier. The prediction accuracy (error rate) of the proposed classifier is calculated through a Monte Carlo simulation method with 100 replications.

1 Introduction

The recording of interval data has become a common practice with the recent advances in database technologies. A natural source of interval data is the aggregation of huge data bases, when real values describing the individual observations result in intervals in the description of the aggregated data. Such type of data have been mainly studied in *Symbolic Data Analysis* (SDA) [1] which is a domain in the area of knowledge discovery and data management, related to multivariate analysis, pattern recognition and artificial intelligence. SDA has been introduced as a new domain related to multivariate analysis, pattern recognition and artificial intelligence in order to extend classical exploratory data analysis and statistical methods to symbolic data. Symbolic data allows multiple (sometimes weighted) values for each variable, and it is why new variable types (interval, categorical multi-valued and modal variables) have been introduced.

Several classifiers has been introduced in SDA. Concerning region oriented approach, Ichino *et al.* [3] introduced a symbolic classifier as a region oriented approach for multi-valued data where the classes of examples are described by a region (or set of regions) obtained through the use of an approximation of a *Mutual Neighbourhood Graph* (*MNG*) and a symbolic join operator. Souza *et al.* [4] proposed a *MNG* approximation to reduce the complexity of the learning step without losing the classifier performance in terms of prediction accuracy and D'Oliveira *et al.* [7] presented a region oriented approach in which each region is defined by the convex hull of the objects belonging to a class. All these region oriented classifiers mentioned above were applied for quantitative data sets and concerning the allocation step, the assignment of a new object to a class is based on a dissimilarity matching function which compares the class description (a region or a set of regions) with a point in $\Re^p$.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 11–18, 2009.

This work addresses a region oriented classifier for symbolic interval data. Here, the region oriented classifier has as input data a symbolic interval data set with known classes and at end of the learning step each class is describe by a region (or a set of regions) in $\Re^p$ defined by the hypercube of the objects belonging to this class. In the allocation step, the assignment of a new object to a class is based on a L_r Minkowski distance between interval feature vectors which compares the class description with the description of a object.

The structure of the paper is as follows: Section 2 presents synthetic interval data sets considered in this work. Section 3 introduces the region oriented classifier for interval data. Section 4 discusses the performance analysis considering two synthetic interval data sets ranging from easy to moderate cases with linearly non-separable clusters. The evaluation of the performance of this non-linear classifier is based on the prediction accuracy. This measurement is assessed in the framework of a Monte Carlo experience with 100 replications of each data set. Section 5 concludes the paper.

2 Symbolic Interval Data

In classical data analysis, the items to be grouped are usually represented as a vector of quantitative or qualitative measurements where each column represents a variable. However, this model is too restrictive to represent complex data, which may, for instance, comprehend variability and/or uncertainty. We may have 'native' interval data, when describing ranges of variable values - for example, daily stock prices. Interval variables also allow to deal with imprecise data, coming from repeated measures or confidence interval estimation.

Suppose that there are K classes. Let C_k $(k = 1, \ldots, K)$ be a class of items labelled $\omega_1, \ldots, \omega_{n_k}$ with $C_k \cap C_{k'} = \emptyset$ if $k \neq k'$ and $\cup_{k=1}^{m} C_k = \Omega$ (data set). Each item i is described by a vector of p symbolic variables $\mathbf{x}_i = (X_1, \ldots, X_p)$. A symbolic variable X_j $(j = 1, \ldots, p)$ is an interval variable when, given an item i of $\Im$, $X_j(i) = x_{ij} = [a_{ij}, b_{ij}] \subseteq \mathcal{A}_j$ where $\mathcal{A}_j = [a, b]$ is an interval.

In this work, two synthetic interval data sets are considered. They are initially generated from two standard quantitative data set in $\Re^2$. These data sets have 450 points scattered among three classes of unequal sizes: one class with ellipse shape of size 200 and two classes with spherical shapes and sizes 150 and 100. Each class in these quantitative data sets were drawn according to a bi-variate normal distribution with non-correlated components and vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ represented by:

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} \text{ and } \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$$

Standard data set 1 showing classes well separated was drawn according to the following parameters:

a) Class 1: $\mu_1 = 50, \mu_2 = 25, \sigma_1^2 = 9, \sigma_2^2 = 36$ and $\rho_{12} = 0.0$;
b) Class 2: $\mu_1 = 45, \mu_2 = -2, \sigma_1^2 = 25, \sigma_2^2 = 25$ and $\rho_{12} = 0.0$;
c) Class 3: $\mu_1 = 38, \mu_2 = 40, \sigma_1^2 = 9, \sigma_2^2 = 9$ and $\rho_{12} = 0.0$;

Standard data set 2 showing overlapping classes was drawn according to the following parameters:

a) Class 1: $\mu_1 = 50$, $\mu_2 = 25$, $\sigma_1^2 = 9$, $\sigma_2^2 = 36$ and $\rho_{12} = 0.0$;
b) Class 2: $\mu_1 = 45$, $\mu_2 = 5$, $\sigma_1^2 = 25$, $\sigma_2^2 = 25$ and $\rho_{12} = 0.0$;
c) Class 3: $\mu_1 = 45$, $\mu_2 = 40$, $\sigma_1^2 = 9$, $\sigma_2^2 = 9$ and $\rho_{12} = 0.0$;

Each data point (z_1, z_2) of each one of this synthetic quantitative data set is a seed of a vector of intervals (rectangle) defined as:$([z_1 - \gamma_1/2, z_1 + \gamma_1/2], [z_2 - \gamma_2/2, z_2 + \gamma_2/2])$. These parameters γ_1, γ_2 are randomly selected from the same predefined interval. The intervals considered in this paper are: $[1, 10], [1, 20], [1, 30]$ and $[1, 40]$. Figure 1 and 2 illustrate these synthetic interval data sets 1 and 2, respectively, ranging from easy to moderate cases of overlapping classes with parameters γ_1 and γ_2 randomly selected from the interval $[1, 10]$.

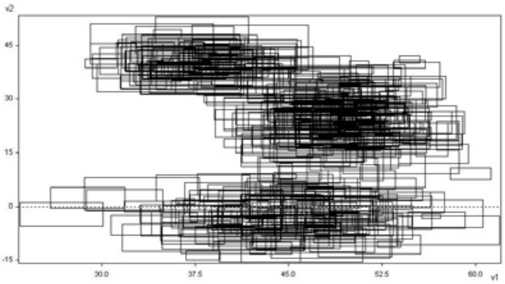


Fig. 1. Symbolic interval data set 1

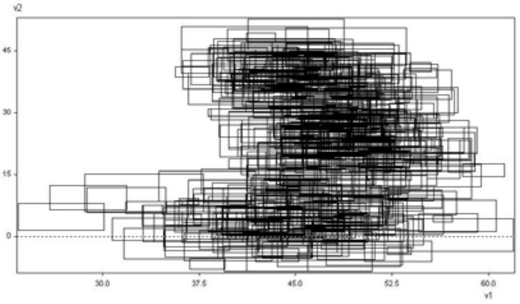


Fig. 2. Symbolic interval data set 2

3 A Region Oriented Classifier for Interval Data

This section describes the learning and allocation steps of a non-linear classifier for symbolic interval data based on a region oriented approach.

3.1 Learning Step

The idea of this step is to provide a description of each class through a region (or a set of regions) in $\Re^p$ defined by the hyper-cubes formed by the objects belonging to this class, which is obtained through a suitable approximation of *MNG*.

Definition 1 Join operator
The join [3] between the interval feature vectors $\mathbf{x}_{ki}$ $(i = 1, \dots, n_k)$ is an interval feature vector which is defined as $\mathbf{y}_k = \mathbf{x}_{k1} \oplus \dots \oplus \mathbf{x}_{kn_k} = (x_{k11} \oplus \dots \oplus x_{kn_k1}, \dots, x_{k1j} \oplus \dots \oplus x_{kn_kj}, \dots, x_{k1p} \oplus \dots \oplus x_{kn_kp})$, where $x_{k1j} \oplus \dots \oplus x_{kn_kj} = [min\{a_{k1j}, \dots, a_{kn_kj}\}, max\{b_{k1j}, \dots, b_{kn_kj}\}](\,j = 1, \dots, p)$.

Definition 2 Join region
The *region* $R(C_k)$ associated to class C_k is a region in $\Re^p$ which is spanned by the join of the objets belonging to class C_k. Its description is given by a interval feature vector $\mathbf{g}(C_k) = (g_{k1}, \dots, g_{kp})$ where $g_{kj} = [\alpha_{kj}, \beta_{kj}]$ is an interval with $\alpha_{kj} = min\{a_{k1j}, \dots, a_{kn_kj}\}$ and $\beta_{kj} = max\{b_{k1j}, \dots, b_{kn_kj}\}$, $(j = 1, \dots, p)$ is an interval value.

Definition 3 Mutual neighborhood graph
The *mutual neighborhood graph (MNG)* [3] yields information about interclass structure. The objects belonging to class C_k are each one *mutual neighbors* [3] if $\forall \omega_{k'i} \in C_{k'}$ $(k' \in \{1, \dots, m\}, k' \neq k), \mathbf{x}_{k'i} \cap \mathbf{g}_k \notin R(C_k)$ $(i = 1, \dots, n_{k'})$. In this case, the *MNG* of C_k against $\overline{C_k} = \cup_{\substack{k' \neq k \\ k'=1}}^{m} C_{k'}$, which is constructed by joining all pairs of objects which are mutual neighbors, is a complete graph.
If the objects belonging to class C_k are not each one mutual neighbors, we look for all the subsets of C_k where its elements are each one mutual neighbors and which are a *maximal clique* in the *MNG*, which, in that case, is not a complete graph. To each of these subsets of C_k we can associate a *region*.

Concerning this step, we have a basic remarks. When the *MNG* of a class C_k is not complete, it is necessary to construct of an approximation of the *MNG* because the computational complexity in time to find all maximal cliques on a graph is exponential.

Algorithm 1 shows the steps for the construction of the *MNG* for the classes C_k $(k = 1, \dots, m)$ and the representation of each class by a *region* (or by a set of *region*). At the end of this algorithm it is computed the subsets $C_k^1, \dots, C_k^{n_k}$ of class C_k and it is obtained the description of this class by the *R-regions* $R(C_k^1), \dots, R(C_k^{n_k})$.

3.2 Allocation Step

The aim of the allocation step is to associate a new object to a class based on a L_r Minkowski distance that compares the class description (an interval vector or a set of interval vectors) with the description of this object (an interval vector).

Let ω be a new object, which is candidate to be assigned to a class $C_k (k = 1, \dots, m)$, and its corresponding description given by the continuous interval feature vector $\mathbf{x}_\omega = (x_{\omega 1}, \dots, x_{\omega p})$ where $x_{\omega j} = [a_{\omega j}, b_{\omega j}]$ is an interval. Remember that from the learning step it is computed the subsets $C_k^1, \dots, C_k^{v_k}$ of C_k.

Algorithm 1. Learning step

```
For each class $k = 1, \ldots, m$ do
 1: Find the the region $R(C_k)$ (according to definition 2) associated to class $C_k$
       Verify if the objects belonging to this class are each one mutual neighbors according to
       the definition 3
 2: If it is the case, construct the MNG (which is a complete graph) and stop
 3: If it is not the case (MNG approximation) do:
    3.1: Choose an object of $C_k$ as a seed according to the lexicographic order of these objects
    in $C_k$;
       Do $t = 1$ and put the seed in the set $C_k^t$;
       Remove the seed from $C_k$
    3.2: Add the next object of $C_k$ (according to the lexicographic order) to $C_k^t$
       If all the objects belonging now to $C_k^t$ remains each one mutual neighbors according to
       the definition 3, remove this object from $C_k$
    3.3: Repeat step 3.2) for all remaining objects in $C_k$
    3.4: Find the region $R(C_k^t)$ (according to definition 2) associated to $C_k^t$)
    3.5: If $C_k \neq \emptyset$,
       Do $t = t + 1$
       Repeat steps 3.1) to 3.4) until $C_k = \emptyset$
    3.6: Construct the MNG (which now is not a complete graph) and stop
```

Algorithm 2. Allocation step

```
For a new example $\omega$ do
 1: For each class $k = 1, \ldots, m$ compute
    $\delta(\omega, C_k) = min\{d_r(\omega, C_k^1), \ldots, d_r(\omega, C_k^{v_k})\}$ where

              $d_r(\omega, C_k^s) = \sum_{j=1}^{p} \phi_r(x_{\omega j}, g_{kj}^s)$        (1)

    $s = 1, \ldots, v_k$ and

              $\phi_r(x_{\omega j}, g_{kj}^s) = [|a_{\omega j} - \alpha_{kj}^s|^r + |b_{\omega j} - \beta_{kj}^s|^r]^{1/r}$        (2)

    with $\mu(*)$ being the range of the interval $*$.
 2: Affect $\omega$ to class $C_k$
              $\delta(\omega, C_k) \leq \delta(\omega, C_h), \forall h \in \{1, \ldots, m\}$
```

The equation (2) corresponds to represent an interval $[a, b]$ as a point $(a, b) \in \Re^2$, where the lower bounds of the intervals are represented in the x-axis, and the upper bounds in the y-axis, and then compute the L_r distance between the points (a_i^j, b_i^j) and (α_i^j, β_i^j). Therefore the distance function in equation (1) is a suitable extension of the L_r metric to interval data. In [5], [6] and [2], respectively, are presented versions of this distance for the cases when $r = 1$ (City-Block), $r = 2$ (Euclidean) and $r = \infty$ (Chebyshev).

4 The Monte Carlo Experiences

Experiments with two synthetic interval data sets in $\Re^2$ showing from easy to moderate cases of classification with linearly non-separable clusters and a corresponding performance analysis of the region oriented classifier introduced in this paper are considered in this section. Monte Carlo experiences with 100 replications of each data set with identical statistical properties are obtained and for each one training (75% of the original data set) and test (25% of the original data set) sets are randomly generated.

The evaluation of the classifier presented in this paper is carried out based on prediction accuracy that was measured through the error rate of classification obtained by the region oriented classifier from a test data set. The estimated error rate of classification corresponds to the average of the error rates found between the 100 replicates of test set.

Table 1 presents the average (in %) and the standard deviation (in parenthesis) of the error rates for interval data sets 1 and 2 as well as γ_1 and γ_2 drawn from [1,10], [1,20], [1,30] and [1,40] according to the L_1, L_2 and L_∞ distances. From the results in this table, we can observe clearly that the L_1 and the L_∞ are, respectively, the best and worst options for both data sets. Moreover, this table shows that the average error rate decreases from the predefined interval $[1, 10]$ to $[1, 20]$. It is why the number of regions in the predefined interval $[1, 20]$ is four times the number of regions in the predefined interval $[1, 10]$.

Table 1. The average (%) and the standard deviation (in parenthesis) of the error rate for synthetic interval data sets 1 and 2

Range of values	Data Set 1			Data Set 2		
of γ_i $i = 1, 2$	L_1 distance	L_2 distance	L_∞ distance	L_1 distance	L_2 distance	L_∞ distance
$[1, 10]$	6.39 (0.0594)	7.47 (0.0636)	9.07 (0.0648)	7.18 (0.0269)	7.11 (0.0278)	7.47 (0.0265)
$[1, 20]$	1.41 (0.0126)	1.40 (0.0121)	1.65 (0.0127)	6.76 (0.0212)	6.58 (0.0221)	6.83 (0.0233)
$[1, 30]$	1.29 (0.0101)	1.27 (0.0099)	1.37 (0.0100)	6.60 (0.0226)	6.49 (0.0226)	6.53 (0.0226)
$[1, 40]$	1.44 (0.0118)	1.35 (0.0120)	1.23 (0.0114)	6.98 (0.0242)	6.77 (0.0244)	6.92 (0.0260)

To compare distances evaluated in this paper, Student's t-test for independent samples at the significance level of 5% was used to determine whether there is a significant difference in error rate between L_1, L_2 and L_∞ distances. In Tables 3 and 4 μ_1, μ_2 and μ_3 are, respectively, the corresponding averages for L_1, L_2 and L_∞ distances. From the values in these tables, we can conclude that, for these interval data sets 1 and 2, the L_2 distance outperforms the L_1 and L_∞ distances in almost all situations.

Table 2. Comparison between the distances for interval data set 1 according to the average error rate

Range of values of γ_i $i = 1, 2$	Hypothesis $H_0 : \mu_1 = \mu_2$ $H_1 : \mu_1 > \mu_2$	Decision	Hypothesis $H_0 : \mu_1 = \mu_3$ $H_1 : \mu_1 < \mu_3$	Decision	Hypothesis $H_0 : \mu_2 = \mu_3$ $H_1 : \mu_2 < \mu_3$	Decision
[1,10]	-124.10	Not Reject H_0	-304.872	Reject H_0	-176.218	Reject H_0
[1,20]	5.724	Reject H_0	-134.153	Reject H_0	-142.520	Reject H_0
[1,30]	14.141	Reject H_0	-56.286	Reject H_0	-71.065	Reject H_0
[1,40]	53.476	Reject H_0	127.991	Not Reject H_0	72.499	Not Reject H_0

Table 3. Comparison between the distances for interval data set 2 according to the average error rate

Range of values of γ_i $i = 1, 2$	Hypothesis $H_0 : \mu_1 = \mu_2$ $H_1 : \mu_1 > \mu_2$	Decision	Hypothesis $H_0 : \mu_1 = \mu_3$ $H_1 : \mu_1 < \mu_3$	Decision	Hypothesis $H_0 : \mu_2 = \mu_3$ $H_1 : \mu_2 < \mu_3$	Decision
[1,10]	18.095	Reject H_0	-76.799	Reject H_0	-93.733	Reject H_0
[1,20]	58.776	Reject H_0	-22.221	Reject H_0	-77.848	Reject H_0
[1,30]	34.416	Reject H_0	21.901	Reject H_0	-12.515	Reject H_0
[1,40]	61.107	Reject H_0	16.892	Not Reject H_0	-42.068	Reject H_0

5 Concluding Remarks

A non-linear classifier for symbolic interval data based on a region oriented approach for interval data was presented in this paper. The input of the region oriented approach is a set of interval feature vectors. Concerning the learning step, each class is described by a region (or a set of regions) in $\Re^p$ defined by the hypercubes formed by the objects belonging to this class, which is obtained through an approximation of a Mutual Neighbourhood Graph (*MNG*). In the allocation step, the assignment of a new object to a class is based on a suitable L_r Minkowski distance between interval feature vectors which compares the class description with the description a of object.

To show the usefulness of the proposed classifier in this paper, two synthetic interval data sets in $\Re^2$ are considered ranging from easy to moderate cases with linearly non-separable clusters. The evaluation was based on prediction accuracy according to the error rate of classification according to the L_1, L_2 and L_∞ distances for interval data. This measurement was calculated in the framework of a Monte Carlo experience with 100 replications.

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

References

1. Bock, H.H., Diday, E.: Analysis of Symbolic Data: Exploratory Methods for Extracting Statistical Information from Complex Data. Springer, Heidelberg (2000)
2. De Carvalho, F.A.T., Brito, P., Bock, H.H.: Dynamic Clustering for Interval Data Based on L2 Distance. Computational statistics (Zeitschrift). Heidelberg (Alemanha) 21, 231–250 (2006)
3. Ichino, M., Yaguchi, H., Diday, E.: Symbolic pattern classifiers based on the cartesian system model. In: Diday, E., et al. (eds.) Ordinal and Symbolic Data Analysis, pp. 92–102. Springer, Berlin (1996)
4. Souza, R.M.C.R., De Carvalho, F.A.T., Frery, A.C.: Symbolic approach to SAR image classification. In: IEEE 1999 International Geoscience and Remote Sensing Symposium, Hamburgo, pp. 1318–1320 (1999)
5. Souza, R.M.C.R., De Carvalho, F.A.T.: Clustering of interval data based on city-block distances. Pattern Recognition Letters 25(3), 353–365 (2004)
6. Souza, R.M.C.R., De Carvalho, F.A.T.: Dynamical clustering of interval data based on adaptive Chebyshev distances. IEE Electronics Letters 40(11), 658–659 (2004)
7. D'Oliveira, S., De Carvalho, F.A.T., de Souza, R.M.C.R.: Classification of SAR images through a convex hull region oriented approach. In: Pal, N.R., Kasabov, N., Mudi, R.K., Pal, S., Parui, S.K. (eds.) ICONIP 2004. LNCS, vol. 3316, pp. 769–774. Springer, Heidelberg (2004)
8. Yaguchi, H., Ichino, M., Diday, E.: A knowledge accquisition system based on the Cartesian space model. In: Diday, E., et al. (eds.) Ordinal and Symbolic Data Analysis, pp. 113–122. Springer, Heidelberg (1996)

A Symmetrical Model Applied to Interval-Valued Data Containing Outliers with Heavy-Tail Distribution

Marco A.O. Domingues, Renata M.C.R. de Souza,
and Francisco José A. Cysneiros

Informatics Center, Federal University of Pernambuco,
P.O.Box 7851 - 50.740-540, Recife (PE), Brazil
{maod,rmcrs}@cin.ufpe.br, cysneiros@de.ufpe.br

Abstract. The aim of Symbolic Data Analysis (SDA) is to provide a set of techniques to summarize large data sets into smaller ones called symbolic data tables. This paper considers a kind of symbolic data called Interval-Valued Data (IVD) which stores data intrinsic variability and/or uncertainty from the original data set. Recent works have been proposed to fit the classic linear regression model to symbolic data. However, those works do not consider the presence of symbolic data outliers. Generally, most specialists treat outliers as errors and discard them. Nevertheless, a single interval-data outlier holds significant information which should not be discarded or ignored. This work introduces a prediction method for IVD based on the symmetrical linear regression (SLR) analysis whose response model is less susceptible to the IVD outliers. The model considers a symmetrical distribution for error which allows to the model possibility of applying regular statistical hypothesis tests.

1 Introduction

Symbolic Data Analysis (SDA) provides a set of techniques that can summarize large data sets into smaller data called symbolic data tables. In a symbolic data table, a cell can contain a distribution, or intervals, or several values linked by a taxonomy and logical rules. In particular, in this paper it will be considered a kind of symbolic data called here as Interval-Valued Data (IVD), which keeps data intrinsic variability and/or uncertainty from the original data set.

SDA could be also defined as the extension of standard data analysis to symbolic data tables. The aim of SDA is to provide suitable methods (e.g., clustering) for managing aggregated data described by multi-value variables, in which the cells of the data table contain sets with categories, intervals or weight (probability) distributions[1].

Recent works have been proposed to fit the classic linear regression model (CLRM) to symbolic data. However, those works do not consider the presence of symbolic data outliers. It is known that CLRM is strongly influenced by outliers. Because the least square predictions are dragged towards the outliers and the

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 19–26, 2009.

variance of the estimates is artificially inflated, the result is that outliers can be masked. Thus, when performing CLRM, most of specialists prefer to discard outliers before computing the line that best fit the data under investigation.

In general, outliers are interpreted as an error. However, a small number of outliers is not due to any anomalous condition and they frequently contain valuable information about the analysed process and should be carefully investigated before being removed.

In the framework of regression models for IVD,[2] presented an approach to fitting a CLRM to IVD sets, which consists of fitting a CLRM to the mid-points of the IVD assumed by the symbolic interval variables. In the following, [3] proposed another approach that fits two independent classic regression models on the lower and upper bounds of the intervals. In [4] the regression methodology is extended to models which include taxonomy predictor variables and models which contain a hierarchical variable structure. Recently, [5] presented the centre and range method for fitting a CLRM to symbolic in which the problem is investigated as an L_2 norm problem and compared this method with the [2] and [3] methods. The cited model do not consider any probabilistic assumptions for the CLRM errors and do not present any treatment for IVD with outliers.

This work introduces a new prediction method for IVD based on the symmetrical linear regression (SLR) analysis. Its innovative feature is that the response model is less susceptible to the presence of IVD outliers. The model considers a heavy-tailed Student-t distribution as an assumption for the errors in the mid-point of the IVD assumed by the symbolic interval variables. The error probability distribution assumption allows to the model possibility of applying regular statistical hypothesis tests, making the model powerful.

The rest of this paper is structured as follows: Section 2 presents the synthetic IVD sets considered in the work. Section 3 describes the SLR models for IVD. Section 4 presents a performance analysis for the proposed method with synthetic IVD sets. Finally, Section 5 gives the concluding remarks.

2 Interval-Valued Data

In the classical data analysis, the items are represented as a vector of quantitative or qualitative measurements in which each column represents a variable. However, this model is too restrictive to represent complex data which may comprehend variability and/or uncertainty. Interval variables permit to deal with imprecise data resulting of repeated measures or confidence interval estimation.

2.1 Synthetic Interval-Valued Data Sets Containing Outlier Rectangles

Synthetic IVD sets in $\Re^2$ are generated from synthetic standard quantitative data sets in $\Re^2$ such that each point belonging to the standard quantitative data set is a centre (seed) for a rectangle in $\Re^2$.

Let I_Y be a response interval variable that is related to a predictor interval variable I_X. Let $E = \{e_1, \ldots, e_n\}$ be an example set where each example e_i $(i = 1, \ldots, n)$ is represented as an interval quantitative feature vector $\mathbf{z} = (I_X(i), I_Y(i))$ with $I_X(i) = [l_X(i), u_X(i)] \in \Im = \{[a,b] : a, b \in \Re, a \leq b\}$ and $I_Y(i) = [l_Y(i), u_Y(i]] \in \Im$.

Let Y^c and X^c be, respectively, standard quantitative variables that assume as their value the mid-point of the interval assumed by the symbolic interval-valued variables I_Y and I_X. Also, let Y^r and X^r be, respectively, quantitative variables that assume as their value the half range of the interval assumed by the symbolic interval-valued variables I_Y and I_X.

Here, four configurations for centre and range according to [5] are considered. These configurations assume that the centre and range of the intervals are simulated independently of uniform distributions.

Let $X^c \sim U[a,b]$ and $X^r \sim U[c,d]$ be, respectively, centre and range variables associated to an independent interval-valued variable I_X. Let $Y^r \sim U[e,f]$ be the range variable associated to the dependent interval-valued variable I_Y. The centre variable Y^c is related to the centre variable X^c as $Y^c = \boldsymbol{\beta_0}+\ \boldsymbol{\beta_1} X^c + \epsilon$, where $\beta_0 \sim U[g,h]$ and $\beta_1 \sim U[g,h]$, and $\epsilon \sim U[l,m]$ is an error.

Each standard data set (in $\Re^2$) has 250 points. Table 1 displays four different configurations with centre and range values to simulate rectangle in $\Re^2$, for $\beta_0 \sim U[-10,10]$ and $\beta_1 \sim U[-10,10]$.

For synthetic IVD sets considered in this paper, a rectangle is an outlier if its mid-point y coordinate is remote in the rectangles set. The effect that this rectangle causes on the regression model depends on the x coordinate of its mid-point and on the general disposition of the other rectangles in the data set.

Outlier IVD are created based on the centre data set $(Y^c(i),\ \mathbf{X}^c(i))$ $(i = 1, \ldots, 250)$. First, this set in $\Re^2$ is sorted ascending by the dependent variable Y^c and a small cluster containing the m first points of the sorted set $(Y^c(i), \mathbf{X}^c(i))$ is selected. The observations of this cluster are changed into outlier points by

$$Y^c(i) = Y^c(i) - 3 * S_{Y^c} \ (i = 1, \ldots, m)$$

where S_{Y^c} is the standard deviation of $Y^c(i)$ $(i = 1, \ldots, 250)$ of data set.

After that, the lower and upper bounds of the intervals $I_X(i)$ and $I_Y(i)$ $(i = 1, \ldots, 250)$ of the rectangle set are obtained by

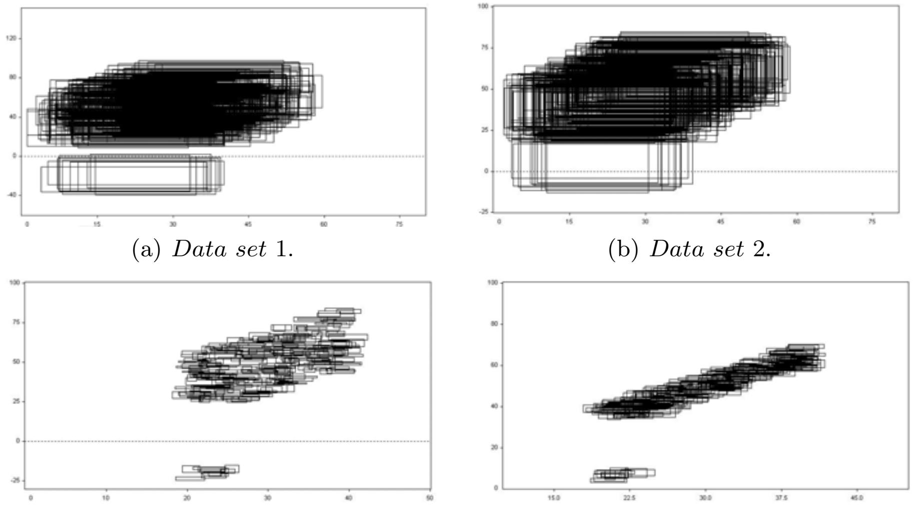
Table 1. Configurations with centre and range for prediction and response variables I_X and I_Y

Config.	Interval-valued variable I_X		Error	Interval-valued variable I_Y	
	Centre	Range		Centre	Range
1	$X^c \sim U[20,40]$	$X^r \sim U[20,40]$	$\epsilon \sim U[-20,20]$	$Y^c = \boldsymbol{\beta_0}+\ \boldsymbol{\beta_1}(\mathbf{X}^c) + \epsilon$	$Y^r \sim U[20,40]$
2	$X^c \sim U[20,40]$	$X^r \sim U[20,40]$	$\epsilon \sim U[-5,5]$	$Y^c = \boldsymbol{\beta_0}+\ \boldsymbol{\beta_1}(\mathbf{X}^c) + \epsilon$	$Y^r \sim U[20,40]$
3	$X^c \sim U[20,40]$	$X^r \sim U[1,5]$	$\epsilon \sim U[-20,20]$	$Y^c = \boldsymbol{\beta_0}+\ \boldsymbol{\beta_1}(\mathbf{X}^c) + \epsilon$	$Y^r \sim U[1,5]$
4	$X^c \sim U[20,40]$	$X^r \sim U[1,5]$	$\epsilon \sim U[-5,5]$	$Y^c = \boldsymbol{\beta_0}+\ \boldsymbol{\beta_1}(\mathbf{X}^c) + \epsilon$	$Y^r \sim U[1,5]$

$$I_X(i) = [X^c(i) - X^r(i)/2, X^c(i) + X^r(i)/2]$$

$$I_Y(i) = [Y^c(i) - Y^r(i)/2, Y^c(i) + Y^r(i)/2]$$

Figures 1 depict the IVD sets 1, 2, 3 and 4. Figures 1(a) and 1(b) describe high variability on the ranges of the rectangles, hence the outlier rectangles are lightly remote in y coordinate, as those figures show. Figures 1(c) and 1(d) show low variability on the ranges of the rectangles, there are outlier rectangles in those figures that are remote in y coordinate.



(a) *Data set* 1.

(b) *Data set* 2.

(c) *Data set* 3.

(d) *Data set* 4.

Fig. 1. Interval-valued data sets 1, 2, 3 and 4 containing outlier rectangles

3 Symmetrical Linear Regression Models

In classic data analysis, the presence of outliers normally affect the regression model. A practical case is one in which the observations follow a distribution that has heavier tails than the normal. These heavy-tailed distributions tend to generate outliers, and these outliers may have a strong influence on the method of least squares in the sense that is no longer an optimal estimation technique[6].

The importance of taking into account the centre (mid-point) and range information in a CLRM for predicting IVD was demonstrated in [5]. In their model, the estimation uses the least square method that does not take into account any probabilistic hypothesis on the response variable. However, this model may also suffer strong influence when there are outlier IVD.

This section presents a SLR model for IVD. The model is less susceptible to the presence of outliers and considers a Student-t distribution as assumption for errors on the mid-point of a learning data set. The assumption that the errors have a probability distribution grants to the model powerful possibilities of applying regular statistical hypothesis tests.

3.1 Symmetrical Linear Regression Models for Classical Data

Suppose $Y_1, \ldots, Y_n$ as n independent random variables where density function is given by

$$f_{Y_i}(y) = \frac{1}{\sqrt{\phi}} g\{(y-\mu_i)^2/\phi\}, \quad y \in I\!R, \tag{1}$$

with $\mu_i \in I\!R$ and $\phi > 0$ being the location and dispersion parameters, respectively. The function $g : I\!R \longrightarrow [0, \infty)$ is such that $\int_0^\infty g(u)du < \infty$ is the density generator and it is denoted by $Y_i \sim S(\mu_i, \phi, g)$.

The SLR model is defined as $Y_i = \mu_i + \epsilon_i, \qquad i = 1, \ldots, n$, where $\mu_i = \mathbf{x}_i^t \boldsymbol{\beta}$, $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_p)^T$ is an unknown parameters vector, additionally, $\epsilon_i \sim S(0, \phi, g)$ and $\mathbf{x}_i$ is the vector of explanatory variables. When they exist, $E(Y_i) = \mu_i = \mathbf{x}_i^t \boldsymbol{\beta}$ and $Var(Y_i) = \xi\phi$, where $\xi > 0$ is a constant that depends on distribution (see, for instance, [7]). This class of models includes all symmetric continuous distributions, such as normal, Student-t, logistic, among others. For example, the Student-t distribution with ν degrees of freedom results in $\xi = \frac{\nu}{\nu-2}$ then $Var(Y_i) = \frac{\nu}{\nu-2}\phi$ and normal distribution $\xi = 1$, $Var(Y_i) = \phi$. In SLR model, $\boldsymbol{\beta} = (\beta_0, \beta_1)^T$ and $\hat{\boldsymbol{\theta}} = (\hat{\boldsymbol{\beta}}^T, \hat{\phi})^t$.

The maximum likelihood estimates of $\boldsymbol{\theta}$, $\hat{\boldsymbol{\theta}} = (\hat{\boldsymbol{\beta}}^T \hat{\phi})^T$ cannot be obtained separately and closed-form expressions for this estimates do not exist. Some iterative procedures can be used such as newton-raphson, BFGS and *scoring Fisher* method. *Scoring Fisher* method can be easily applied to get $\hat{\boldsymbol{\theta}}$ where the process for $\hat{\boldsymbol{\beta}}$ can be interpreted as a weighting least square. The iterative process for $\hat{\boldsymbol{\theta}}$ takes the form

$$\boldsymbol{\beta}^{(m+1)} = \{\mathbf{X}\mathbf{D}(\mathbf{v}^{(m)})\mathbf{X}\}^{-1}\mathbf{X}^t\mathbf{D}(\mathbf{v}^{(m)})\boldsymbol{y}. \tag{2}$$

$$\phi^{(m+1)} = \frac{1}{n}\{\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}\}^T\mathbf{D}(\mathbf{v})\{\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}\} \qquad (m = 0, 1, 2, \ldots). \tag{3}$$

with $\mathbf{D}(\mathbf{v}) = \text{diag}\{v_1, \ldots, v_n\}$, $\boldsymbol{y} = (y_1, \ldots, y_n)^t$, $\mathbf{X} = (\mathbf{x}_1^t, \ldots, \mathbf{x}_n^t)^t$ and $v_i = -2\mathrm{W}_g(u_i)$, $\mathrm{W}_g(u) = \frac{g'(u)}{g(u)}$, $g'(u) = \frac{dg(u)}{du}$ and $u_i = (y_i - \mu_i)^2/\phi$. For the normal distribution the maximum likelihood estimates take closed-form expressions, because $v_i = 1, \forall i$. For the Student-t distribution with ν degrees of freedoms, we have $g(u) = c(1 + u/\nu)^{-(\nu+1)/2}, \nu > 0$ and $u > 0$ so that $\mathrm{W}_g(u_i) = -(\nu+1)/2(\nu+u_i)$ and $v_i = (\nu+1)/(\nu+u_i), \forall i$. In this case the current weight $v_i^{(r)}$ from (2) is inversely proportional to the distance between the observed value y_i and its current predicted value $\mathbf{x}_i^t\boldsymbol{\beta}^{(r)}$, so that outlying observations tend to have small weights in the estimation process (see discussion, for instance, in [8]).

3.2 Construction of the Symmetrical Linear Regression Models for Interval-Valued Data

In this model, each example e_i $(i = 1, \ldots, n)$ is represented by two vectors $\mathbf{z}_c = (X^c(i), Y^c(i))$ and $\mathbf{z}_r = (X^r(i), Y^r(i))$ where $X^c(i) = [l_X(i) + u_X(i)]/2$, $X^r(i) = [u_X(i) - l_X(i)]/2$, $Y^c(i) = [l_Y(i) + u_Y(i)]/2$ and $Y^r(i) = [u_Y(i) - l_Y(i)]/2$.

The predictor variable I_X is related to the response variable I_Y according to two linear regression equations, respectively, on their centre and range values

$$\begin{aligned} Y^c &= \beta_0^c + \beta_1^c X^c \\ Y^r &= \beta_0^r + \beta_1^r X^r \end{aligned} \tag{4}$$

The parameters β_0^c and β_1^c are estimated by the maximum likelihood method according to subsection 3.1 and the parameters β_0^r and β_1^r are estimated by the least squared method [6].

3.3 Rule of Prediction

The prediction of the lower and upper bounds $\hat{I}_Y(v) = [\hat{l}_Y, \hat{u}_Y]$ of a new example v is based on the prediction of $\hat{Y}^c(v)$ and $\hat{Y}^r(v)$. Given the interval $I_X(v) = [l_X, u_X]$ with $X^c(v) = (l_X + u_X)/2$ and $X^r(v) = (u_X - l_X)/2$, the interval $\hat{I}_Y(v) = [\hat{l}_Y, \hat{u}_Y]$ is obtained as follow:

$$\hat{l}_Y = \hat{Y}^c(v) - \hat{Y}^r(v) \text{ and } \hat{u}_Y = \hat{Y}^c(v) - \hat{Y}^r(v)$$

$$\text{where} \quad \hat{Y}^c(v) = \hat{\beta}_0^c(v) + \hat{\beta}_1^c X^c(v) \text{ and } \hat{Y}^r(v) = \hat{\beta}_0^r(v) + \hat{\beta}_1^r X^r(v)$$

4 Performance Analysis

In order to demonstrate the usefulness of the symmetrical model presented in this paper, experiments with four synthetic IVD sets in $\Re^2$ are considered in this section. These IVD sets contain outlier rectangles. Moreover, the proposed SLR model is compared with the linear regression model for IVD introduced by [5].

4.1 Results for Synthetic Interval Data Sets

Here, the analysis was performed in the framework of a Monte Carlo experience with 100 replications of each data set. Test and learning sets are randomly selected from each synthetic IVD set. The learning set corresponds to 75% of the original data set and the test data set corresponds to 25%.

The performance assessment of the SLR model presented is based on the *pooled root mean-square error* ($PRMSE$). This measure is obtained from the observed interval values $I_Y(i) = [l_Y(i), u_Y(i)]$ $(i = 1, \ldots, n)$ of I_Y and from their corresponding predicted interval values $\hat{I}_Y(i) = [\hat{l}_Y(i), \hat{u}_Y(i)]$ and it is estimated in the framework of a Monte Carlo simulation with 100 replications in two ways.

For each learning synthetic IVD set, the $PRMSE$ measure is given by

$$PRMSE^1 = \sqrt{\frac{\sum_{i=1}^{250} \omega(i) error(i)}{250}} \qquad \text{where,}$$

$$error(i) = [(l_Y(i) - \hat{l}_Y(i))^2 + (u_Y(i) - \hat{u}_Y(i))^2]$$

and $\omega(i)$ is the weight of the residual $r_i = Y^c(i) - \hat{Y}^c(i)$ $(i = 1, \ldots, 250)$ obtained from symmetrical model described in subsection 3.1 and adopted to fit $(Y^c(i), \mathbf{X}^c(i))$. In the centre and range model [5], the least squares criterion function weights all residuals equally to 1.0.

For each test synthetic IVD set, the $PRMSE$ measure is given by

$$PRMSE^2 = \sqrt{\frac{\sum_{i=1}^{125} error(i)}{125}}$$

The $PRMSE^1$ and $PRMSE^2$ measures are estimated for each fixed configuration. At each replication of the Monte Carlo method, a SLR model to the learning synthetic IVD set is fitted. Thus, the fitted model is used to predict the interval values of the dependent interval-valued variable I_Y in the test and learning synthetic IVD sets.

For each $PRMSE^k$ $(k = 1, 2)$, the average and standard deviation over the 100 Monte Carlo simulations are calculated and a statistical Student's t-test for paired samples at a significance level of 1% is then applied to compare the SLR model proposed in this paper to the linear regression model for IVD introduced by [5]. The null and alternative hypotheses are, respectively:

$$H_0 : (PRMSE^k)^{Symmetrical} = (PRMSE^k)^{Linear}$$
$$H_1 : (PRMSE^k)^{symmetrical} < (PRMSE^k)^{Linear}.$$

where $(PRMSE^k)^{Symmetrical}$ and $(PRMSE^k)^{Linear}$ are, respectively, the *pooled root mean-square error* for symmetrical model and *pooled root mean-square error* for proposed model in [5].

Table 2 displays the ratio of times that the hypothesis H_0 is rejected for the measures $PRMSE^1$ and $PRMSE^2$ regarding all configurations of IVD sets.

Table 2. Comparison between regression models according to the rejection ratio (%) of H_0 for $PRMSE^1$ and $PRMSE^2$

Config.	$PRMSE^1$	$PRMSE^2$
1	100	100
2	100	100
3	100	100
4	100	100

The results in the table above show clearly that the SLR model for IVD is superior to the regression model proposed in [5]. For all test data sets in this evaluation the rejection ratios of H_0 are equal to 100%.

5 Conclusions

A symmetrical linear prediction model for symbolic IVD is introduced in this paper. The input data set is described by feature vectors, for which each feature is an interval. The relationship between a dependent interval variable (response variable) and an independent interval variable is modeled by information contained in the range and mid-point of the intervals. A symmetrical linear prediction is fitted for mid-points and the prediction of the lower and upper bounds of the intervals is performed from these fits. A linear regression is fitted for range.

In order to validate the introduced symmetrical model for IVD, experiments with synthetic IVD sets containing outlier IVD are considered. The fit and prediction qualities are assessed by on a pooled root mean square error calculated from learning and test data sets, respectively, and the results provided by the proposed method are compared to the correspondent results provided by regression model for interval data presented in [5]. For synthetic IVD set, this measure is estimated in the framework of Monte Carlo simulations.

The comparison between the models is achieved by hypothesis tests at 1% level of significance. Regarding the pooled root mean square error, the results showed that the symmetrical model is superior to centre-range model in terms of fit and prediction qualities. This fact points out that, according to this evaluation, the introduced symmetrical linear model is not sensitive in the presence of outlier interval-valued data.

The next step is to investigate the behavior of the symmetrical model assuming different families of heavy-tailed probability distribution for the errors.

References

1. Diday, E., Noirhomme-Fraiture, M.: Symbolic Data Analysis and the SODAS Software. Wiley, West Sussex (2008)
2. Billard, L., Diday, E.: Regression Analysis for Interval-Valued Data. In: 7th Conf. of Int. Fed. of Classif. Soc., pp. 369–374. Springer, Belgium (2000)
3. Billard, L., Diday, E.: Symbolic Regression Analysis. In: 8th Conf. of Int. Fed. of Classif. Soc., pp. 281–288. Springer, Poland (2002)
4. Billard, L., Diday, E.: Symbolic Data Analysis: Conceptual Statistics and Data Mining. Wiley, West Sussex (2006)
5. Lima Neto, E.A., De Carvalho, F.A.T.: Centre and Range method for fitting a linear regression model to symbolic interval data. CSDA 52, 1500–1515 (2008)
6. Montgomery, D.C., Peck, E.A.: Introduction to Linear Regression Analysis. Wiley, New York (1982)
7. Fang, K.T., Kotz, S., Ng, K.W.: Symmetric Multivariate and Related Distributions. Chapman and Hall, London (1990)
8. Cysneiros, F.J.A., Paula, G.A.: Restricted methods in symmetrical linear regression models. Comp. Stat. and Data Analysis 49, 689–708 (2005)

New Neuron Model for Blind Source Separation

Md. Shiblee[1], B. Chandra[2,⋆], and P.K. Kalra[1]

[1] Department of Electrical Engineering
Indian Institute of Technology Kanpur, India
{shiblee,kalra}@iitk.ac.in

[2] Department of Industrial Engineering and Managment
Indian Institute of Technology Kanpur, India
bchandra104@yahoo.co.in

Abstract. The paper proposes new neuron model with an aggregation function based on Generalized harmonic mean of the inputs. Information-maximization approach has been used for training the new neuron model. The paper focuss on illustrating the efficiency of the proposed neuron model for blind source separation. It has been shown on various generated mixtures (for blind source separation) that the new neuron model performs far superior compared to the conventional neuron model.

1 Introduction

Blind Source Separation (BSS) using neural network is one of the most emerging areas in the field of neural network. Blind source separation (BSS) refers to the problem of recovering statistically independent signals from a linear mixture[1]. The application of neural base blind source separation to extract signals of independent sources from their linear mixture has already been considered in several real life problem such as in telecommunications, biometric, EGG, signal and image processing. BSS using conventional neuron model with information maximization approach has been attempted by several authors [2,3,4]. The performance of these algorithms is usually influenced by the selection of the neuron model. The drawback of the existing models is that non-linearity can not be captured accurately. In this paper, we have purposed new neuron model which is based on generalized harmonic mean (GHMN). The conventional perceptron model is the special case of this neuron model. The order of hyperplane in generalized harmonic mean based neuron (GHMN) model is higher than that of conventional model and thus the GHMN captures nonlinearity more efficiently. Information-maximization approach has been used as a learning algorithm in the GHMN model. The performance of the information-maximization approach using GHMN model has been evaluated on a numbers of generated blind source mixtures and compared with a performance of the conventional neuron model and EASY algorithm for independent algorithm given by David Gleich [5]. The results reveal the superiority of the GHMN model.

[⋆] Permanent Add: Department of Mathematics, IIT Delhi, India.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 27–36, 2009.

Rest of the paper is organized as follows. In section 2, the new neuron model is presented. Section 3 gives an overview of blind source separation problem. The training algorithm of new neuron model for the blind source separation is discussed in Section 4. In section 5, results have been tested on several generated mixtures of signals. In section 6, we have concluded our work.

2 Generalized Harmonic Mean Neuron Model

Neuron modeling concerns with relating function to the structure of the neuron on the basis of its operation. The MLP neuron is based on the concept of weighted arithmetic mean of the N input signals

$$Weighted\ arithmetic\ mean\ (x_1, x_2, \cdots, x_N) = \frac{1}{N}\sum_{i=1}^{N} w_i.x_i \tag{1}$$

The higher order neurons models are difficult to train because of a combinatorial explosion of higher order terms as the number of inputs to the neuron increases. To overcome this problem, as the name suggests we proposed generalized harmonic mean neuron (GHMN) model which is based on the concept of generalized harmonic mean. The generalized harmonic mean of the N inputs can be found by the summing operation as follows [6]:

$$Mean\ (x_1, x_2, \cdots, x_N) = \left(\frac{N}{\sum_{i=1}^{N} {x_i}^p}\right)^{-1/p} \tag{2}$$

$Case\ 1 : p = -\infty$

$$\begin{aligned} Mean\ (x_1, x_2, \cdots, x_N) &= \lim_{p \to -\infty}\left(\frac{N}{\sum_{i=1}^{N} {x_i}^p}\right)^{-1/p} \\ &= Min(x_i) = Min\ operation \end{aligned} \tag{3}$$

$Case\ 2 : p = -1$

$$Mean\ (x_1, x_2, \cdots, x_N) = \left(\frac{N}{\sum_{i=1}^{N} {x_i}^{-1}}\right)^{1}$$

$$= \left(\frac{N}{\sum_{i=1}^{N} \frac{1}{x_i}} \right) = Harmonic\ mean \tag{4}$$

$Case\ 3 : p = 0$

$$Mean\ (x_1, x_2, \cdots, x_N) = \lim_{p \to 0} \left(\frac{N}{\sum_{i=1}^{N} {x_i}^p} \right)^{-1/p}$$

$$= (x_1, x_2, x_3, \cdots, x_N)^{\frac{1}{N}} = Geometric\ mean \tag{5}$$

$Case\ 4 : p = 1$

$$Mean\ (x_1, x_2, \cdots, x_N) = \frac{1}{N} \sum_{i=1}^{N} x_i = Arithmetic\ mean \tag{6}$$

$Case\ 5 : p = 2$

$$Mean\ (x_1, x_2, \cdots, x_N) = \left(\frac{N}{\sum_{i=1}^{N} {x_i}^2} \right)^{-1/2}$$

$$= \frac{1}{N} \left(\sum_{i=1}^{N} x_i^2 \right)^{1/2} = Quadratic\ mean \tag{7}$$

$Case\ 6 : p = \infty$

$$Mean\ (x_1, x_2, \cdots, x_N) = \lim_{p \to \infty} \left(\frac{N}{\sum_{i=1}^{N} {x_i}^p} \right)^{-1/p}$$

$$= Max(x_i) = Max\ operation \tag{8}$$

The aggregation function of the new neuron model which gives the weighted generalized harmonic mean of the N input signals of the neuron is defined as

$$nety = \left(\frac{1}{\sum_{i=1}^{N} w_i \cdot {x_i}^p} \right)^{-1/p} + b \tag{9}$$

where w_i is adaptive parameter corresponding to input x_i . b is the bias of the neuron. $nety$ is the net output passing trough the activation function.

On passing net value through an activation function φ , the output y will be given as:

$$y = \varphi(nety) \tag{10}$$

From Eq. 10 , we find that for p=1

$$y = \varphi\left(\sum_{i=1}^{N} w_i \cdot x_i + b\right) \tag{11}$$

which is the output of conventional neuron model [7].

3 Blind Source Separation

Blind source separation (BSS) problems are those in which several source signals are mixed together and the objective is to extract the original source signals from the mixtures [1]. The best example of BSS is Cocktail Party Problem, in which more than one person start talking at a time and the listener is unable to judge any of the conversations. The term blind in BSS is used because the mixing process and sometimes number of source signals are not known. Several statistical and neural approaches have been purposed for the solution of this problem. Independent component analysis (ICA) is one of the methods which is most widely used for blind source separation. In terms of neural network, ICA can be viewed as an unsupervised technique which tries to represent data in terms of statistically independent variables. In this section the basic data model of ICA has been defined [8,9].

Consider a source vector S.

$$S = [S_1, S_2, \cdots, S_N]^T$$

where N is the number of mutually statistically independent source. The vector S is input to a linear system whose input-output characterization is defined by nonsingular $N - by - N$ mixing matrix A. The result is a $N - by - 1$ mixture vector X related to S as follows:

$$X = A \times S$$

where

$$X = [X_1, X_2, \cdots, X_N]^T$$

In real life situation both the source vector S and mixing matrix A are unknown. The objective is to find de-mixing matrix W from mixture vector X such that the source vector S can be recovered from output vector Y defined as $S = W \times X$ in Eq.12 as:

$$Y = [Y_1, Y_2, \cdots, Y_N]^T \tag{12}$$

Here, source vector S, mixing vector X and output vector Y are normalized between zero to one. In this paper we are using logistic transform function which has the range from 0 to 1 and due to use of power term; it is restriction that the scaled value of X must be positive.

4 Learning Algorithm

For blind source separation, we first expect the outputs of the separation system to be statistically independent. For this purpose, we must utilize a measure of independence. As mutual information is the best index to measure of independence so in this paper mutual information-maximization approach has been used for learning of the purposed neuron model for blind source separation. The mutual information that the output Y of a neural network contains about its input X can be defined as [10,11]:

$$I(Y, X) = H(Y) - H(Y/X) \tag{13}$$

where $H(Y)$ is the entropy of the output and $H(Y/X)$ is entropy which does not come from input. Thus in this case, $H(Y/X)$ is the part which does not depend on weight [12]. To maximize the mutual information, the differentiation of the equation (1) with respect to $'W'$ can be written as:

$$\frac{\partial}{\partial W} I(Y, X) = \frac{\partial}{\partial W} H(Y) \tag{14}$$

$\frac{\partial}{\partial W} H(Y/X) = 0$, Thus the maximization of the mutual information $I(Y/X)$ is equivalent to maximization of the output entropy $H(Y)$.

Bell and Sejnowski [2] stated that when we pass a single input x through a transforming function $g(x)$ to give an output variable y, both $I(y, x)$ and $H(y)$ are maximized when we align high density parts of the probability density function (pdf) of x with highly sloping parts of the function $g(x)$. This is the idea of "matching a neuron's input-output function to the expected distribution of single" that we find in [13]

When $g(x)$ monotonically increases or decreases, the pdf of the output,$f_y(y)$, can be written as a function of the pdf of the input, $f_x(x)$,[14]

$$f_y(y) = \frac{f_x(x)}{|\partial y / \partial x|} \tag{15}$$

The entropy of the output $H(y)$ is given by [10]:
$H(y) = -E[lnf_y(y)] = -\int_{-\infty}^{\infty} f_y(y)\ lnf_y(y)$

$$= E\left[\ln|\frac{\partial y}{\partial x}|\right] - E[ln\ f_x(x)] \tag{16}$$

In order to maximize the entropy of y by changing w and bias b, we need only concentrate on maximizing the first term, which is the average log of how the input affects the output. The second term on the right (the entropy of x) may be considered to be unaffected by alterations in a parameter w and b determining $g(x)$. The weight update equations using gradient descent learning rule are given below:

$$\Delta w \propto \frac{\partial H}{\partial w} = \frac{\partial}{\partial w}\left(\ln|\frac{\partial y}{\partial x}|\right) = \left(\frac{\partial y}{\partial x}\right)^{-1} \frac{\partial}{\partial w}\left(\frac{\partial y}{\partial x}\right) \tag{17}$$

$$\Delta b \propto \frac{\partial H}{\partial b} = \frac{\partial}{\partial b}\left(\ln|\frac{\partial y}{\partial x}|\right) = \left(\frac{\partial y}{\partial x}\right)^{-1}\frac{\partial}{\partial b}\left(\frac{\partial y}{\partial x}\right) \tag{18}$$

In the case of logistic transfer function: $y = \frac{1}{1+e^{-u}}$, $u = \left(\frac{1}{\sum_{i=1}^{N} w_i \cdot {x_i}^p}\right)^{-1/p}$

where w is the weight and x is the input and b is the bias of the neuron. Then the partial differentiation of output with respect to input can be written as

$$\frac{\partial y}{\partial x} = y(1-y)\left(w \cdot x^p\right)^{\frac{1}{p}-1} w \cdot x^{p-1} \tag{19}$$

and

$$\Delta w = \left(\frac{\partial y}{\partial x}\right)^{-1}\frac{\partial}{\partial w}\left(\frac{\partial y}{\partial x}\right) \tag{20}$$

$$\Delta w \propto \left[\frac{1}{w}\left[\left(1-\frac{1}{p}\right)\cdot\frac{1}{(w \cdot x^p)} + (1-2y)\left(w \cdot x^p\right)^{\frac{1}{p}-1}\right]\right]\cdot x^p \tag{21}$$

similarly

$$\Delta b = \left(\frac{\partial y}{\partial x}\right)^{-1}\frac{\partial}{\partial b}\left(\frac{\partial y}{\partial x}\right) \tag{22}$$

$$\Delta b = (1-2y) \tag{23}$$

The above equation can be used as a weight update rule for the case of an input and one output.

Putting $q = 1$ gives:

$$\Delta w = \frac{1}{w} + x(1-2y) \tag{24}$$

which is the weight update rule given by Bell and Sejnowski for conventional neuron model.

Let us consider a multidimensional inputs and outputs network with an input vector X, a weight matrix W, a bias vector b and a monotonically transformed output vector Y. weight update rule can be extended for this as:

$$\Delta W \propto \frac{\partial H}{\partial W} = \frac{\partial}{\partial W}\left(\ln|\frac{\partial Y}{\partial X}|\right) = \left(\frac{\partial Y}{\partial X}\right)^{-1}\frac{\partial}{\partial W}\left(\frac{\partial Y}{\partial X}\right) \tag{25}$$

For our proposed model with sigmoidal units $Y = g(U), \;\; U = \left(\frac{1}{W \cdot X^p}\right)^{-1/p}$

With g being the logistic function:$y = g(\frac{1}{1+e^{-u}})$ the resulting learning rule is familiar in form:

$$\varDelta W \propto \left[\frac{1}{W}\left[\left(1-\frac{1}{p}\right)\cdot\frac{1}{(W\cdot X^p)}+(1-2Y)\,(W\cdot X^p)^{\frac{1}{p}-1}\right]\right]\cdot X^p \qquad (26)$$

$$\varDelta b = (1-2Y) \qquad (27)$$

5 Results

The results were tested on various generated mixtures of signals. The input signal is normalized between 0-1. Logistic transformation is used to limit the output in the range of 0-1. In all the BSS problems all the diagonal elements of the initial weight matrix has chosen to be unity. The result is compared with the BSS with information maximization approach using classical model and EASY algorithm for independent algorithm given by David Gleich [5].

5.1 Example 1

The sources used to generate mixtures are sawtooth wave and Sin wave. Following Fig. 1(a) show the wave forms of mixture.

$$x_1 = 9 \cdot sawtooth(5a) + 4 \cdot sin(5a)$$
$$x_2 = 3 \cdot sawtooth(5a) + 3 \cdot sin(5a)$$

Figs 1(b) and Figs 1(c) show the separated signal with information maximization approach using GHMN model and classical model respectively and fig.1(d) shows the separated signal by EASY-ICA algorithm. From the results it is visible that BSS using GHMN model separate, Sin and sawtooth wave more accurately than BSS using classical neuron model and EASY-ICA algorithm. And BSS using classical neuron model with information maximization approach performs better than EASY-ICA algorithm.

5.2 Example 2

Three mixtures are synthetically generated with trigonometric, algebraic and logarithmic functions. Sin, wave is taken as trigonometric function, square wave is taken as algebraic and log wave is taken as logarithmic function. The mathematical equations used to generate mixtures are given below:

$$x_1 = 2 \cdot log(2a) + 4 \cdot square(5a) + 9 \cdot sin(3a)$$
$$x_2 = 8 \cdot log(2a) + 3 \cdot square(5a) + 3 \cdot sin(3a)$$
$$x_3 = 3 \cdot log(2a) + 7 \cdot square(5a) + 2 \cdot sin(3a)$$

wave forms of mixtures are given in Fig. 2(a). The separated signals from the mixture using various model has been depicted in Figs 2(b), (c) and (d) respectively. The inference drawn for the earlier example holds true for this example also. Since BSS using GHMN model outperform other two models.

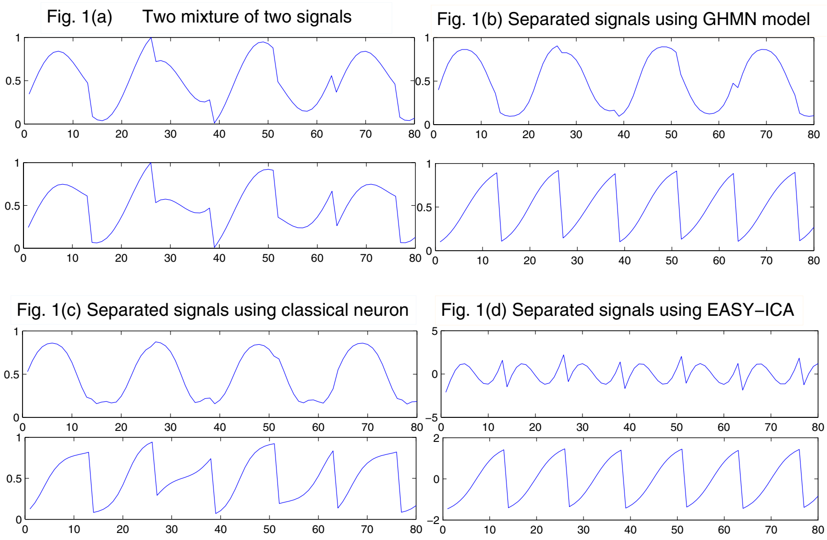


Fig. 1(a) Two mixture of two signals

Fig. 1(b) Separated signals using GHMN model

Fig. 1(c) Separated signals using classical neuron

Fig. 1(d) Separated signals using EASY−ICA

Fig. 1.

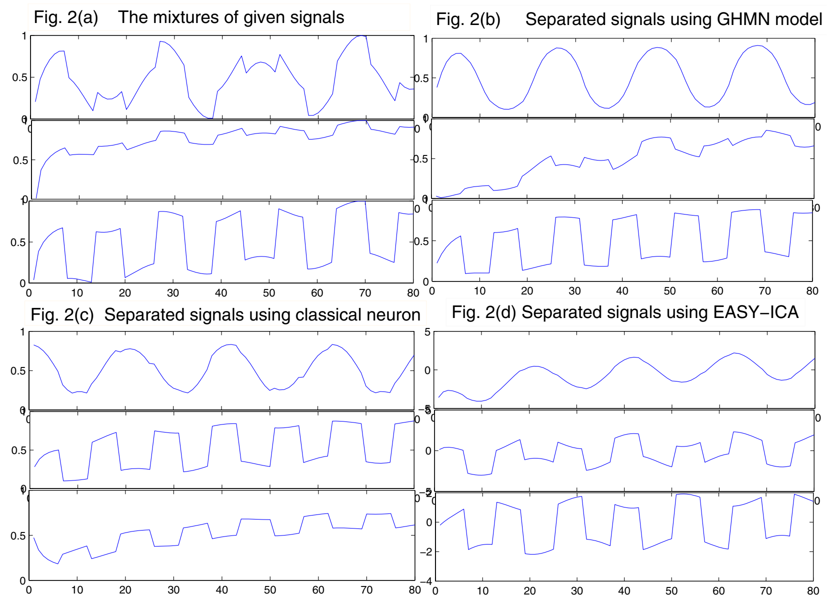


Fig. 2(a) The mixtures of given signals

Fig. 2(b) Separated signals using GHMN model

Fig. 2(c) Separated signals using classical neuron

Fig. 2(d) Separated signals using EASY−ICA

Fig. 2.

5.3 Example 3

Three mixtures of different harmonic are generated using following equations:

$$
\begin{aligned}
x_1 &= 2 \cdot sin(160a) + 4 \cdot [sin(60a) + 6 \cdot cos(60a)] + 9 \cdot sin(cos(180a)) \\
x_2 &= 8 \cdot sin(160a) + 3 \cdot [sin(60a) + 6 \cdot cos(60a)] + 3 \cdot sin(cos(180a)) \\
x_1 &= 3 \cdot sin(160a) + 7 \cdot [sin(60a) + 6 \cdot cos(60a)] + 2 \cdot sin(cos(180a))
\end{aligned}
$$

The wave forms of mixtures are given in Fig. 3(a). The separated signals from the mixture using various model and EASY-ICA algorithm has been depicted in Figs 3(b), (c) and (d) respectively. The inference drawn for the earlier example 1 and 2 holds true for this example also.

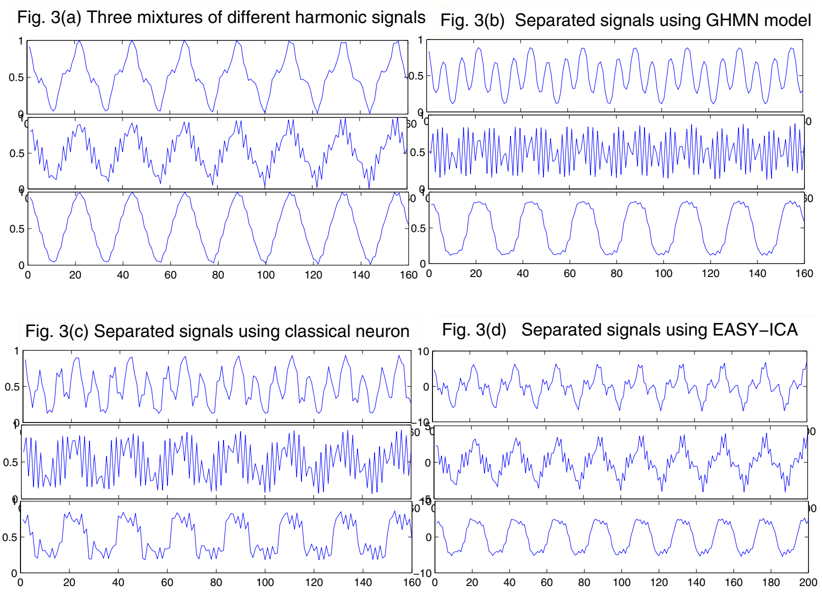


Fig. 3(a) Three mixtures of different harmonic signals

Fig. 3(b) Separated signals using GHMN model

Fig. 3(c) Separated signals using classical neuron

Fig. 3(d) Separated signals using EASY−ICA

Fig. 3.

6 Conclusions

This paper proposes a new neuron model base on generalized harmonic mean information-maximization has been used as the learning algorithm. Comparative performance evaluations of this propose model depicts that separation of original sources in the blind source mixtures is far superior compare to conventional neuron model.

References

1. Hyvarinen, A., Karhunen, J., Oja, E.: Indepndent Component Analysis-Theory and Applications. John Wiley & Sons, New York (2001)
2. Bell, A.J., Sejnowsi, T.J.: An information-maximization approach to blind separation and blind deconvolution. Neural Computation 7, 1129–1159 (1995)

3. Hyvarinen, A., Oja, E.: Independent component analysis: algorithms and application. Neural Networks 13, 411–430 (2000)
4. Lee, T.W.: Blind source separation of nonlinear mixing models. Neural Networks 7, 121–131 (1997)
5. http://www.stanford.edu/~dgleich/publications/pca-neural-nets-website
6. Kijima, M.: The generalized harmonic mean and a portfolio problem with dependent assets. Theory and decision 43, 71–87 (1997)
7. Minsky, M.I., Papert, S.A.: Perceptrons. MIT Press, Cambridge (1969)
8. Jutten, C., Herault, J.: Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture. Signal Processing 24, 1–10 (1991)
9. Comon, P.: Independent component analysis- a new concept? Signal Processing 36, 287–314 (1994)
10. Cover, T.M., Thomas, J.A.: Elements of information theory. Wiley, New York (1991)
11. Haykin, S.: Neural networks: a comprehensive foundation. MacMillan, New York (1994b)
12. Nadal, J.-P., Parga, N.: Non-linear neurons in the low noise limit: a factorial code maximizes information transfer. Networks 5, 565–581 (1994)
13. Laughlin, S.: A simple coding procedure enhances a neuron's information capacity. Z. Naturforsch 36, 910–912 (1981)
14. Papoulis, A.: Probability, random variables and stochastic processes, 2nd edn. McGraw-Hill, New York (1984)
15. Cover, T.M., Thomas, J.A.: Elements of information theory. Wiley, New york (1991)

Time Series Prediction with Multilayer Perceptron (MLP): A New Generalized Error Based Approach

Md. Shiblee[1], P.K. Kalra[1], and B. Chandra[2,⋆]

[1] Department of Electrical Engineering
Indian Institute of Technology Kanpur, India
[2] Department of Industrial Engineering and Managment
Indian Institute of Technology Kanpur, India
{shiblee,kalra}@iitk.ac.in, bchandra104@yahoo.co.in

Abstract. The paper aims at training multilayer perceptron with different new error measures. Traditionally in MLP, Least Mean Square error (LMSE) based on Euclidean distance measure is used. However Euclidean distance measure is optimal distance metric for Gaussian distribution. Often in real life situations, data does not follow the Gaussian distribution. In such a case, one has to resort to error measures other than LMSE which are based on different distance metrics [7,8]. It has been illustrated in this paper on wide variety of well known time series prediction problems that generalized geometric and harmonic error measures perform better than LMSE for wide class of problems.

1 Introduction

Time series prediction and forecasting are key problems of function approximation. In the existing literature, various neural network learning algorithms have been used for these problems [1,2,3]. In this paper, we have used multilayer perceptron (MLP) neural network for time series prediction. MLP is composed of a hierarchy of processing units, organized in series of two or more mutually exclusive sets of neurons or layers. The input layer serves as the holding site for the input applied to the network. The output layer is the point at which the overall mapping of the network input is available [4,5]. Most widely used algorithm for learning MLP is the back-propagation algorithm [6]. In Back-propagation algorithm, error between target value and observed value is minimized. Typically Euclidean distance is used to for the error measure. It has been proven based on Maximum Likelihood criterion that Euclidean distance is optimal distance metric for Gaussian distribution [7,8].Since the distribution of data is unknown, using LMSE for training of MLP may not give the actual approximation of the functions.In this paper, Back-propagation algorithm with some new error measures based on distance metrics (for similarity measures) given by Jie Yu et al

[⋆] Permanent Add: Department of Mathematics, IIT Delhi, India.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 37–44, 2009.

[7,8] has been used for training MLP for time series prediction. Comparative performance for each of the error measures has been done on a wide variety of well known time series prediction problems. The performance measures used are chi-square goodness of fit test, AIC and training and testing error.

Rest of the paper is organized as follows, Section 2, gives the overview of distance metrics used as various error measures. The network architecture of feed-forward (MLP) neural network is presented in section 3. In section 4, learning rule of MLP using Back-propagation (BP) algorithm with different error measures has been discussed. In section 5, results have been tested on several widely known time series prediction and forecasting data sets. In section 6, we have concluded our work.

2 New Error Measures

Traditionally Back-propagation (BP) algorithm is used in training Artificial Neural Network (ANN). The performance of Neural Network depends on the method of error computation. It has been proven that when the underlying distribution of data is Gaussian, Least mean square error (LMSE) [7,8] is best for training the network. %begincenter Since the distribution of data is unknown, it may be possible that error computed based on Euclidean distance may not be suitable for training of neural network. It will be reasonable to assume that there may be some distance metric which will give new error measure to fit the unknown data better. Some new error measures based on distance metric for similarity measures [7,8] is given in Table 1. In Table 1, y_i denotes the desired value of neuron, t_i denotes the target value and N denotes number of pattern.In error estimation, if the target value is far away from the desired value, error measure

Table 1.

	Error Measure
LMS	$E = \frac{1}{2}\sum_{i=1}^{N}(t_i - y_i)^2$
Geometric	$E = \frac{1}{2}\sum_{i=1}^{N}[log(t_i/y_i)]^2$
Harmonic	$E = \frac{1}{2}\sum_{i=1}^{N}[t_i(y_i/t_i) - 1]^2$
Generalized Geometric	$E = \frac{1}{2}\sum_{i=1}^{N}[{t_i}^r \log(t_i/y_i)]^2$
Generalized Harmonic Type-1	$E = \frac{1}{2}\sum_{i=1}^{N}[{t_i}^p(y_i/t_i) - 1]^2$
Generalized Harmonic Type-2	$E = \frac{1}{2}\sum_{i=1}^{N}({t_i}^q - {y_i}^q)^2$

based on the geometric and harmonic distance metric will be lower in comparison to least mean square error (LMSE). Geometric and Harmonic errors are less sensitive to outliers. Hence the training of neural network using geometric and harmonic error is more robust.

In Generalized Geometric and Harmonic errors, parameters r, p and q define specific error measures. When $r = 0$, generalized geometric error becomes simple geometric error, and when $p = 1$, $q = -1$, both types of Generalized Harmonic errors become simple harmonic error and for $p = 2$ and $q = 1$, both give least mean square error (LMSE) [8].

3 Multi-Layer Perceptron

Multilayer Perceptron (MLP) (Fig.1) consists of an input layer of source nodes, one or more than one hidden layers of neurons and an output layer [4,5]. The number of nodes in the input and the output layers depends on the number of input and output variables, respectively. The input signal propagates through the network layer-by-layer [9].

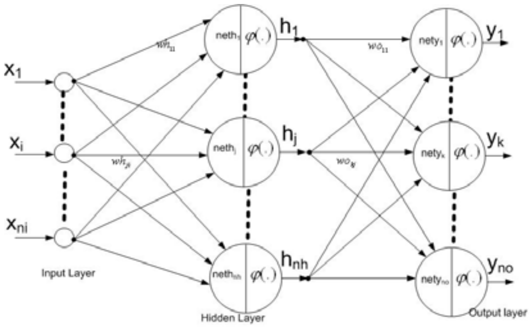


Fig. 1.

It has been proved that a single hidden layer is sufficient to approximate any continuous function [11]. A three layer MLP is thus taken into account in this paper. Computation of the output of MLP carried out as follows [5]:

The net value at j^{th} neuron of hidden layer is

$$neth_j = \sum_{i=1}^{ni} (wh_{ji}.x_i + bh_j),\ j = 1, 2, \ldots, nh \tag{1}$$

where x_i is the input at i^{th} node of input layer.

wh_{ji} is the connection weight of i^{th} neuron with j^{th} input.
bh_j is the weight of the bias at j^{th} neuron of hidden layer.
ni is the number of neuron in input layer.

Output of i^{th} neuron of hidden layer is

$$h_j = \phi(neth_j) = \frac{1}{1 + e^{-neth_j}} \tag{2}$$

The net value at k^{th} neuron of output layer is

$$nety_k = \sum_{j=1}^{nh} (wo_{kj}.h_j + bo_k), \; k = 1, 2, \ldots, no \tag{3}$$

where h_j is the output of j^{th} neuron of hidden layer.

wo_{kj} is the connection weight of j^{th} hidden layer neuron with k_{th} output layer neuron.
bo_k is the weight of the bias at k^{th} neuron of output layer.
nh is the number of neurons in hidden layer.

Output of k_{th} neuron of output layer is $y_k = \phi(nety_k) = \frac{1}{1+e^{-nety_k}}$.

4 Training Algorithm of MLP

We describe an error back-propagation learning rule for the training of the network with new error measures. The weight equations using gradient descent rule are given bellow:

$$wh_{ji}(new) = wh_{ji}(old) + \eta \cdot \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial wh_{ji}} \tag{4}$$

$$bh_j(new) = bh_j(old) + \eta \cdot \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial bh_j} \tag{5}$$

$$wo_{kj}(new) = wo_{kj}(old) + \eta \cdot \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial wo_{kj}} \tag{6}$$

$$bo_k(new) = bo_k(old) + \eta \cdot \frac{\partial E}{\partial y_k} \cdot \frac{\partial y_k}{\partial bh_k} \tag{7}$$

where, η is the learning rate and

$$\frac{\partial y_k}{\partial wh_{ji}} = [\sum_{k=1}^{no} (1 - y_k) \cdot y_k \cdot w_{kj}] \cdot (1 - h_j) \cdot h_j \cdot x_i \tag{8}$$

$$\frac{\partial y_k}{\partial bh_j} = [\sum_{k=1}^{no} (1 - y_k) \cdot y_k \cdot w_{kj}] \cdot (1 - h_j) \cdot h_j \tag{9}$$

$$\frac{\partial y_k}{\partial wo_{kj}} = (1 - y_k) \cdot y_k \cdot h_j \tag{10}$$

$$\frac{\partial y_k}{\partial bo_k} = (1 - y_k) \cdot y_k \tag{11}$$

For different error criterion only $\frac{\partial E}{\partial y_k}$ will change. This is dependent on the distance used in computation of the total error E.
Case 1: Euclidean distance, $LMSE = E = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{no}(t_k - y_k)^2$

$$Then,\ \frac{\partial E}{\partial y_k} = -\eta \cdot (t_k - y_k) \tag{12}$$

Case 2: Geometric distance, $E = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{no}[log(y_k/t_k)]^2$

$$Then,\ \frac{\partial E}{\partial y_k} = -\eta \cdot (log(t_k) - log(y_k)) \cdot 1/y_k \tag{13}$$

Case 3: Harmonic distance, $E = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{no} t_k \cdot (y_k/t_k - 1)^2$

$$Then,\ \frac{\partial E}{\partial y_k} = \eta \cdot (t_k) \cdot (y_k/t_k - 1) \tag{14}$$

Case 4: Generalized geometric distance, $E = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{no}[{t_k}^r \cdot log(t_k/y_k - 1)]^2$

$$Then,\ \frac{\partial E}{\partial y_k} = -\eta \cdot {t_k}^{2r} \cdot (log(t_k) - log(y_k)) \cdot 1/y_k \tag{15}$$

Case 5: Generalized harmonic distance type 1, $E = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{no} {t_k}^p \cdot (y_k/t_k - 1)^2$

$$Then,\ \frac{\partial E}{\partial y_k} = \eta \cdot {t_k}^p \cdot (y_k/t_k - 1) \tag{16}$$

Case 6: Generalized harmonic distance type 2, $E = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{no}({t_k}^q - {y_k}^q)^2$

$$Then,\ \frac{\partial E}{\partial y_k} = -\eta \cdot q \cdot ({t_k}^q - {y_k}^q) \cdot {y_k}^q \tag{17}$$

5 Results

Performance of different error measures using MLP was tested on a number of widely known time series prediction data sets. The data in all the datasets were normalized between 0.1 to 1.0.For the sake of comparison, learning rate and number of epochs was kept same for all errors. Average sum squared error alone may not accurately reflect the accuracy of the network. Thus accuracy of the network is evaluated based on chi-square goodness of fit test, Akaiki information criterion (AIC) and training and testing error [11,12,13].

5.1 Short-Term Internet Incoming Traffic

Short-term Internet traffic data was supplied by HCL-Infinet (a leading Indian ISP). We propose a model for predicting the Internet traffic using previous values. Four measurements at time t, (t-1), (t-2) and (t-4) were used to predict the incoming internet traffic at time (t+1). The structure of the model has four neurons in the hidden layer. In this case, out of 300, 70% samples are taken for training and 30% for testing. To train the network a learning rate of 0.2 was chosen and the number of epochs was taken as 1000.

5.2 Financial Time Series

In time series prediction to predict the future index of stock is a great challenge. For prediction 77 data points (August 28 1972- December 18 1972) [14] were chosen, 50 data sets were taken for training and 27 data sets were taken for testing purpose. Previous two indices were used to predict future index. Three neurons were used in the hidden layer. Regarding the learning parameters 2000 epochs was chosen with learning rate of 0.01.

5.3 Petroleum Sales

Petroleum sales data for the period Jan 1971 to Dec 1991 [14] was used for prediction of sales. Sales of petrol in the previous three months were taken to predict the sale in the fourth month. The data comprises of 252 points. For training of the network, 196 data were used and 53 data points were taken for testing. In the hidden layer four neurons were used. Training of the network is done till 2000 epochs with a learning rate of 0.02.

5.4 Box-Jenkis Gas Furnace

The Box-Jenkins gas furnace data [15] set was recorded from a combustion process in which air and methane were combined in order to obtain a mixture of gases which contained CO2. We modeled the furnace output as a function of the previous output y(t-1) and input x(t-1). The training was performed on 145 samples and 145 samples were used for testing. In the hidden layer four neurons were taken. Network is trained till 2000 epochs with a learning rate of 0.02.

5.5 Australian Monthly Electricity Production

Australian monthly electricity production data was used for forecasting [14]. 371 data points were used in training and 98 data points were used for testing. Data in the periods (t-1), (t-2), (t-3) was used for predicting for time period (t). The structure of model contains four neurons in the hidden layer. Learning parameters are same as chosen for the previous two data set.

5.6 Cow Milk Production

Monthly milk production per cow over 14 years [14] was used for time series prediction. Data in the previous two periods was used for prediction. Till one thousand epochs network was trained with a learning rate of 0.02. In the network, the hidden layer comprised of three neurons.

Table 2.

	LMS	Geometric	Harmonic	Generalized Geometric	Generalized Harmonic Type 1	Generalized Harmonic Type 2
HCL						
TRE	0.0033	0.0031	0.0033	0.0033	0.0032	0.0032
TTE	0.0043	0.0040	0.0043	0.0043	0.0042	0.0041
CHI	2.2254	2.1484	2.1893	2.1778	2.1756	2.0793
AIC	-19.64	-19.76	-19.60	-19.70	-19.65	-19.55
DOWJONES						
TRE	9.1133e-004	9.8006e-004	6.9288e-004	7.4659e-004	7.4022e-004	6.4087e-004
TTE	0.0012	0.0012	9.3175e-004	8.9355e-004	9.8927e-004	8.6135e-004
CHI	0.3685	0.3354	0.3179	0.3078	0.3152	0.2960
AIC	-3.32	-3.28	-3.45	-3.42	-3.42	-3.49
PETROL						
TRE	8.4186e-004	4.7920e-004	5.8087e-004	5.8497e-004	5.5775e-004	4.6957e-004
TTE	0.0068	0.0031	0.0038	0.0028	0.0038	0.0031
CHI	1.1819	0.8223	0.8874	0.7899	0.9259	0.8151
AIC	–13.55	-14.66	-14.28	-14.70	-14.27	-14.36
GAS FURNACE						
TRE	3.2815e-004	4.4026e-004	3.6901e-004	3.3292e-004	3.4909e-004	3.4125e-004
TTE	0.0016	0.0017	0.0012	0.0014	0.0014	0.0012
CHI	0.9319	0.9051	0.8460	0.8272	0.8546	0.8041
AIC	-10.39	-10.96	-11.22	-11.37	-11.30	-11.39
ELECTRICAL						
TRE	7.4763e-004	7.4891e-004	6.7267e-004	6.0223e-004	6.9835e-004	7.1438e-004
TTE	0.0084	0.0081	0.0079	0.0079	0.0078	0.0078
CHI	1.9166	1.8710	1.8279	1.8048	1.8531	1.8319
AIC	-26.38	-26.77	-26.39	-27.18	-26.64	-26.55
COW MILK						
1-7 TRE	0.0056	0.0052	0.0056	0.0051	0.0053	0.0049
TTE	0.0201	0.0188	0.0204	0.0177	0.0188	0.0182
CHI	2.2254	2.1484	2.1893	2.1778	2.1756	2.0793
AIC	-5.97	-6.07	-5.97	-6.10	-6.04	-6.06

It is seen from Table 2 that in all the data sets, the training and testing error was much lower for generalized errors in comparison to LMSE. For the data sets HCL, Dow Jones, Gas furnace and Electrical, Generalized harmonic performs best in terms of testing error. The chi-square values and AIC for these data sets also justify that generalized errors perform best. For Petrol and Cow milk Generalized geometric performs the best in terms of testing error and chi-square test and AIC.

6 Conclusions

In this paper, a novel approach of using different error measures in MLP is suggested. The weight update equations have been evaluated for each of the error measures. It has been shown on several widely known time series prediction and forecasting problems that error measures other than LMSE gives much better performance.

References

1. Dudul, S.V.: Prediction of a Lorenz choaotic attractor using two-layer perceptron neural network. Appl. Soft Computing 5(4), 333–355 (2003)
2. Carpinteiro, O.A.S., Reis, A.J.R., da Silva, A.A.P.: A hierarchical neural model in short-term load forecasting. Appl. Soft Computing 4(4), 405–412 (2004)
3. Xu, K., Xie, M., Tang, L.C., Ho, S.L.: Application of neural networks in forecasting engine systems reliability. Appl. Soft Computing 2(4), 255–268 (2003)
4. Hornik, K., Stinchombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural networks 2, 359–366 (1989)
5. Zurada, J.M.: Introduction to Artificial Neural Systems. Jaicob publishing house, India (2002)
6. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error back propogation. In: Parallel Distributed Processing, vol. 1, MIT Press, Cambridge (1986)
7. Yu, J., Amores, J., Sebe, N., Tian, Q.: Toward an improve Error metric. In: Proc. 2004 ICIP International Conference on Image Processing (2004)
8. Yu, J., Amores, J., Sebe, N., Tian, Q.: Toward Robust Distance Metric Analysis for Similarity Estimation. In: Proc. 2006 IEEE Computer Sosciety Conference on Computer Vision and Pattern Recognition (2006)
9. Haykin, S.: Neural Networks: A comprehensive Foundation. Pearson Education, Singapore (2003)
10. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems 2, 303–314 (1989)
11. Greenwood, P.E., Nikulin, M.S.: A guide to chi-squared testing. Wiley, New York (1996)
12. Akaike, H.: A new look at the statistical model identification. IEEE Tran. Appl. Comp. AC-19, 716–723 (1974)
13. Frogel, D.B.: An information criterion for optimal neural network selection. IEEE Tran. Neural Network 2, 490–497 (1991)
14. Makridakis, Wheelwright, Hyndman: Forecasting: Methods and Applications, 3rd edn. John Wiley and Sons, Chichester (1998)
15. Box, G.E.P., Jenkins, G.M., Reinse, G.C.: Times Series Analysis: Forecasting and control. Prentice Hall, Englewood Cliffs (1994)

Local Feature Selection in Text Clustering

Marcelo N. Ribeiro[1], Manoel J.R. Neto[2], and Ricardo B.C. Prudêncio[1]

[1] Centro de Informática, Universidade Federal de Pernambuco, Recife – PE – Brazil
{mnr,rbcp}@cin.ufpe.br

[2] Instituto de Computação, Universidade Federal de Alagoas, Maceió – AL – Brazil
mjrn@tci.ufal.br

Abstract. Feature selection has improved the performance of text clustering. Global feature selection tries to identify a single subset of features which are relevant to all clusters. However, the clustering process might be improved by considering different subsets of features for locally describing each cluster. In this work, we introduce the method ZOOM-IN to perform local feature selection for partitional hierarchical clustering of text collections. The proposed method explores the diversity of clusters generated by the hierarchical algorithm, selecting a variable number of features according to the size of the clusters. Experiments were conducted on Reuters collection, by evaluating the bisecting K-means algorithm with both global and local approaches to feature selection. The results of the experiments showed an improvement in clustering performance with the use of the proposed local method.

1 Introduction

Clustering algorithms have been applied to support the information access in large collections of textual documents [7]. Such techniques may organize similar documents in clusters (groups) associated to different levels of specificity and different contexts. The structure of clusters, properly labeled, offers a vision of what types of questions can be answered by the query results.

In order to accomplish the text clustering process, the documents are represented, in most cases, as a set of *indexing terms* associated to numerical weights. Considering all existing terms in a collection brings some difficulties to the clustering algorithm. In fact, when the size of the feature space is very high, the distance between similar points is not very different than the distance between more distant points (i.e.,"curse of dimensionality") [8].

Considering the above context, text clustering usually contains a phase of dimensionality reduction of the vectors that represent the documents. Features in the reduced space may correspond to a subset of the original features (as performed by feature selection methods [2]), or they may be created by combining the original features (as performed by feature extraction methods [8]). In text clustering, feature extraction presents a disadvantage compared to feature selection since each new feature is no longer associated with an existing term or word, which makes the formed clusters less comprehensive [8].

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 45–52, 2009.

Feature selection can be classified as either global or local [5]. The global approach aims to select a single subset of features which are relevant to *all* derived clusters [5]. Despite the large use of global methods in literature, depending on the problem, it is possible that there are several different subsets of features that show good clusters. In order to overcome this limitation, local feature selection, in turn, tries to identify different subsets of features associated to each formed cluster. Although recent work has obtained good empirical results by evaluating local feature selection on non-textual data (see [5]), there is no investigation of the use of local feature selection for text clustering.

In this work, we proposed the ZOOM-IN, a local feature selection method for partitional hierarchical clustering. In this method, all the documents are initially allocated to a single top-level cluster which is recursively divided into small subclusters. At each division step, a feature selection criterion is applied to choose the features which are more relevant only considering the cluster being divided. The number of selected features is defined according to the cluster size. The result of our method is a hierarchy of clusters in which each cluster is represented by a different subset of features.

Experiments were performed on the Reuters collection [4], comparing the bisecting K-means algorithm (a partitional hierarchical algorithm) [7], with both the global and local feature selection approaches. The results revealed an improvement in precision when the local approach was compared to the global approach. The ZOOM-IN method eliminated irrelevant terms, at same time maintaining the quantity of information required for each division of the clusters.

Section 2 brings a brief introduction to text clustering. Section 3 presents feature selection approaches applied to text clustering, followed by Section 4 which presents the proposed method. Section 5 brings the experiments and results. Finally, Section 6 presents some final considerations and future work.

2 Text Clustering

Text clustering is the process of grouping similar documents into clusters, in order to better discriminate documents belonging to different categories. A document in text clustering is described by a set of keywords, so-called *terms of indexing*, which is a vocabulary extracted in the collection of texts. A weight is associated to each term, defining an array of terms that represents the document. The term weights are commonly computed by deploying the Vector Space Model with the TF-IDF weighting schema. In this model, each term weight $tfidf_j$ in the document d_j is given by:

$$tfidf_j = tf_j \log \frac{n}{DF_t} \tag{1}$$

where n is the number of documents in the collection, DF_t is the number of documents in the corpus where the term t occurs and tf_j is the frequency of the term t in the document d_j. The proximity between two document vectors $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$, represented in this model, is usually defined by the *cosine* measure as:

$$\cos(\boldsymbol{d}_1, \boldsymbol{d}_2) = \frac{\boldsymbol{d}_1 \cdot \boldsymbol{d}_2}{\|\boldsymbol{d}_1\| \times \|\boldsymbol{d}_2\|} \tag{2}$$

A hierarchical clustering algorithm groups the data in a hierarchy of clusters. For text clustering, the hierarchical solution has more advantages regarding a flat approach, since it divides the collection of documents on various levels of granularity and specificity, providing a better view of the collection.

The hierarchical clustering algorithms may be further categorized into agglomerative or partitional. Agglomerative clustering is a bottom-up approach which starts affecting each document to a distinct cluster and progressively joins similar clusters. Partitional clustering, in turn, is a top-down approach which starts with all documents in a single cluster and progressively divides the existing clusters. In the agglomerative clustering, wrong decisions of combining clusters at the beginning of the algorithm execution tend to multiply errors as the clustering is executed. Partitional algorithms, in turn, have a more global vision of possible cohesive clusters, and hence, they will be the focus of our work. A widespread partitional algorithm is the bisecting K-means [7], in which the simple K-means algorithm is used to bisect the clusters (i.e., dividing each cluster in two subclusters) at each division step. The bisecting K-means has shown to be very competitive compared to agglomerative algorithms [7].

3 Feature Selection for Text Clustering

Feature selection for text clustering is the task of disregarding irrelevant and redundant terms in the vectors that represent the documents, aiming to find the smallest subset of terms that reveals "natural" clusters of documents [2]. Using a small subset of relevant terms will speed up the clustering process, while avoiding the curse of dimensionality. The methods commonly used to select features in text clustering deploy statistical properties of the data as a criterion to determine the quality of the terms [1,8] (see Section 3.1). The selection is performed with the use of a threshold or a fixed number of desired features.

3.1 Criteria for Ranking Features

In this section, we cited some criteria which will be later applied in our experiments:

Document Frequency (DF). The value DF_t of the term t is defined as the number of documents in which the term t occurs at least once in the collection of documents.

Term Frequency Variance (TfV). Let tf_j be the frequency of term t in the document d_j. The quality of term t is defined in the TfV method as:

$$TfV_t = \sum_{j}^{n} tf_j^2 - \frac{1}{n}\left[\sum_{j}^{n} tf_j\right]^2 \tag{3}$$

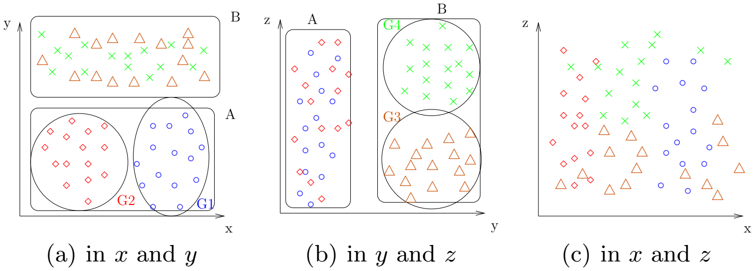
where n is the number of documents in the collection. In the experiments performed in [1], the TfV method has maintained the precision of the clustering process with up to 15% of the total number of features.

Mean of TF_IDF (TI). In [8], the quality of a term t is defined as the mean value of $tfidf_j$ across all documents ($j = 1, \ldots, n$) in the collection. The TI method has shown a performance superior to DF and similar to TfV [8].

3.2 Global and Local Feature Selection

Feature selection for clustering may occur on either the global or the local approach. The global feature selection chooses the relevant features once by deploying a pre-defined ranking criterion, and uses the same subset of features in the whole clustering process. Global selection is the most investigated approach in the literature [2,8]. In local feature selection, a subset of features is chosen for each cluster. It assumes that the clusters may be better discriminated from each other by considering a different subset of features for each cluster.

Figure 1 illustrates a set of objects belonging to four clusters, which are described by the features x, y and z. The clusters G1 and G2 are only revealed when the attributes x and y are considered, i.e., the attribute z is irrelevant to distinguish between G1 and G2 (see Figure 1(a)). Figure 1(b), in turn, illustrates that features y and z are relevant to identify the clusters G3 and G4, i.e., feature x is irrelevant in this context. Finally, the attributes x and z correspond to an irrelevant subset of features (Figure 1(c)). In such situation, any subset of features eventually returned by a global method would not be able to identify the four existing clusters. It is necessary to examine a feature in the context of different subsets before stating that the feature is actually irrelevant [2].



(a) in x and y (b) in y and z (c) in x and z

Fig. 1. Data of the clusters G1, G2, G3 and G4 for different features

Compared to the global approach, there is few relevant work in the literature of clustering that investigated the local feature selection. In [5], for instance, the authors proposed a local feature selection method for clustering, by searching several subsets of features that show different clusters, and choosing the most cohesive clustering based on a criterion of cluster evaluation. In [5], experiments were performed to evaluate the proposed local method for the K-means algorithm. By using the local method, the authors obtained an improvement in precision for clustering benchmarking problems from the UCI repository. We highlight here that, to the best of our knowledge, the local approach for feature selection has not been applied in any work for text clustering.

4 Proposed Method

In this work, it is proposed an algorithm to perform partitional hierarchical clustering with local feature selection. In our proposal, it is expected that the privileged global vision of partitional algorithms can take advantage of a local vision offered by the local feature selection. We also expected that the variety of subsets of features selected to each division of the clusters might reveal hidden clusters in the data. The proposed algorithm for local feature selection using the bisecting K-means follows the steps:

1. Choose a cluster to divide, considering an initial cluster containing all the documents;
2. Select features for the chosen cluster by deploying a ranking criterion (as cited in Section 3.1). The features may be filter based on a pre-defined number of N required terms or based on a threshold τ on the ranking criterion;
3. Build 2 sub-clusters using the K-means algorithm;
4. Repeat steps 2 and 3 by $ITER$ times;
5. Repeat steps 1, 2, 3 and 4 until the required number of clusters is reached.

The problem in Figure 1 can be initially solved by selecting the subset of features (e.g., x and y) that best reveals clusters in the data (e.g., Figure 1(a)). Following, the algorithm generates sub-clusters to both clusters A (the data of G1 plus G2) and B (the data of G3 plus G4). By performing a new feature selection to each cluster, the cluster A remains with the features x and y and cluster B with the features y and z. The cluster A can now be broken into G1 and G2 (children of the cluster A). Cluster B, in turn, can be broken into G3 and G4 (children of the cluster B), thus revealing all clusters for these data.

An important aspect to be considered in our algorithm is the number N of terms to be selected for each cluster. As the algorithm is executed, the generated clusters become smaller, and hence, the number of distinct terms in documents also decreases. Thus, the choice of a large constant number N tends to minor the selection potential (capacity to select all the relevant features) of our method since the number of selected terms will be similar to the number of distinct terms. The choice of a small constant number N, on other hand, will cause a lost of information when the clusters are large.

A solution to the above trade-off is to use a variable number of terms according to the size of the clusters and the number of distinct terms. For simplicity, in our work, it is proposed to choose the number of terms n_i for the cluster i as:

$$n_i = \left\lfloor \frac{N_T}{N_C} \cdot m_i \right\rfloor \tag{4}$$

where N_T is the number of different terms in the collection of documents, N_C is the size of the collection of documents and m_i is the size of the cluster i. N_T/N_C is the proportion of different terms revealed in each document of the collection. This procedure reduces the number of terms locally selected in each division of cluster. Because this reminds the setting of a binocular, this method is referred in this work as *ZOOM-IN* method. As it will be seen, we performed experiments with both the local feature selection with constant number of features and the ZOOM-IN to decide the number of selection terms per each iteration. This method is similar to that proposed in [3] to text categorization, being said as "glocal", since it performs *global* feature selection at each split of clusters and it is *local* because it makes feature selection to the clusters resulting from a split, in such a way that each set of sibling nodes of the dendogram is represented by a different subset of features.

5 Experiments and Results

Section 5.1 describes the experiments performed to evaluate the viability of the proposed method. Section 5.2, in turn, presents the obtained results.

5.1 Experiments Description

In our experiments, we used a subset of documents in the Reuters-21578 collection [4] which were assigned to a single class (representing a total number of 1228 documents associated to 42 classes). The collected documents were initially processed in order to remove stopwords (prepositions and common words). We also applied the stemming operator with the Porter's algorithm.

The clustering algorithms were evaluated by deploying the *micro-averaged precision* measure, also used, for instance, in [6]. The micro-averaged precision assumes that each cluster formed by the clustering algorithm has a majority representative class c. Considering T the set of clusters and C the set of classes, the micro-averaged precision is given by [6]:

$$P\left(T\right) = \frac{\sum_{c \in C} \alpha\left(c, T\right)}{\sum_{c \in C} \alpha\left(c, T\right) + \beta\left(c, T\right)} \tag{5}$$

where $\alpha\left(c, T\right)$ is the number of documents correctly affected to c and $\beta\left(c, T\right)$ is the number of documents incorrectly affected to c.

For evaluating the hierarchy generated by the clustering algorithms, the clusters considered for computing the precision were those present in the leaves of the produced dendogram and the number of clusters specified for execution of the algorithms was equal to the number of classes of the collection. Finally, the micro-average precision was averaged over 30 different runs of the evaluated algorithms.

5.2 Results and Discussion

Figure 2 presents the precision obtained for each evaluated algorithm (bisecting K-means with both the global and local feature selection), with constant number of selected features. As it may be seen, for all global methods, when few terms are selected the performance of the global methods falls. The criterion of ranking that obtained best precision rates is the TfV, with performance similar to TI and better than the DF, as already observed in the work [8]. It is concluded that for a few number of selected terms, there is little information on the documents, which deteriorates the precision of the clustering.

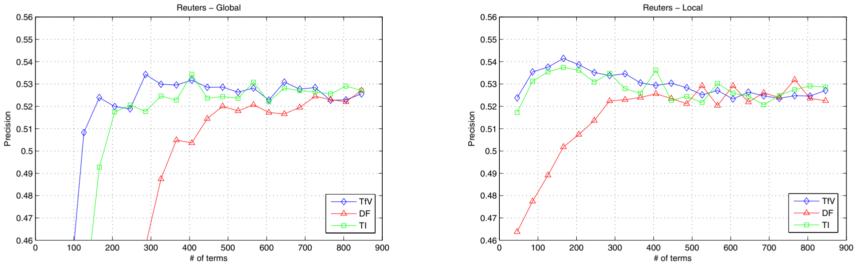


Fig. 2. Micro-averaged precision in relation to the number of terms used for the Reuters collection, with local and global feature selection

The local approach, in turn, keeps the precision even with a very small quantity of terms, excepting for the method DF. Interestingly, the precision obtained for fewer selected terms is even better than the precision obtained when a large number of features is selected. This is due to the fact that for large values of the number of selected terms, the selective potential decreases locally (as discussed in Section 4). With a small amount of selected terms, the selective potential is kept during the divisions of the clusters, and the precision is improved.

Table 1. Micro-averaged precision with the ZOOM-IN method

Collection	Without method	TfV	DF	TI
Reuters	0.527117	0.540988	0.527362	0.541395

However, a small amount of selected terms may undermine the amount of information needed at the beginning of the execution of the clustering, when the clusters are still large and the number of distinct terms as well. It is necessary to select the terms that reflect a real benefit to the clustering. In this context, we also performed an experiment using a variable amount of locally selected terms, which is called the ZOOM-IN method (as proposed in Section 4). The values of precision obtained by ZOOM-IN were even better that the results obtained by the local method with few features (see Table 1).

6 Conclusions

In this work, it was proposed the use of a local feature selection approach for partitional hierarchical text clustering. Each set of sibling nodes derived by the proposed method is represented by a different subset of features. In the performed experiments, the local approach was compared to the global feature selection approach for the bisecting K-means. It was observed that the local approach obtained good precision even for few selected terms. We also performed experiments by using the ZOOM-IN method to automatically define the number of selected features in each iteration of the partitional algorithm. The results obtained by the ZOOM-IN were satisfactory, because it showed the benefits in locally selecting features.

As future work, we intend to evaluate other criteria for ranking features, which use information from the similarity between documents, such as the ranking based on entropy. Finally, the proposed method will be evaluated on other collections of documents.

References

1. Dhillon, I., Kogan, J., Nicholas, C.: Feature selection and document clustering. In: Berry, M.W. (ed.) Survey of Text Mining, pp. 73–100 (2003)
2. Dy, J.G., Brodley, C.E.: Feature selection for unsupervised learning. Journal of Machine Learning Research 5, 845–889 (2004)
3. Koller, D., Sahami, M.: Hierarchically classifying documents using very few words. In: ICML 1997: Proceedings of the Fourteenth International Conference on Machine Learning, pp. 170–178 (1997)
4. Lewis, D.D.: Reuters-21578 text categorization test collection distribution 1.0 (1999), http://www.daviddlewis.com
5. Li, Y., Dong, M., Hua, J.: Localized feature selection for clustering. Pattern Recognition Letters 29(1), 10–18 (2008)
6. Slonim, N., Friedman, N., Tishby, N.: Unsupervised document classification using sequential information maximization. In: Proceedings of the 25th International ACM SIGIR Conference, pp. 129–136 (2002)
7. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. Technical Report, Department of Computer Science and Engineering, University of Minnesota (2000)
8. Tang, B., Shepherd, M., Milios, E., Heywood, M.I.: Comparing and combining dimension reduction techniques for efficient text clustering. In: International Workshop on Feature Selection for Data Mining (2005)

Sprinkled Latent Semantic Indexing for Text Classification with Background Knowledge

Haiqin Yang and Irwin King

Department of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, New Territories, Hong Kong
{hqyang,king}@cse.cuhk.edu.hk

Abstract. In text classification, one key problem is its inherent dichotomy of polysemy and synonym; the other problem is the insufficient usage of abundant useful, but unlabeled text documents. Targeting on solving these problems, we incorporate a sprinkling Latent Semantic Indexing (LSI) with background knowledge for text classification. The motivation comes from: 1) LSI is a popular technique for information retrieval and it also succeeds in text classification solving the problem of polysemy and synonym; 2) By fusing the sprinkling terms and unlabeled terms, our method not only considers the class relationship, but also explores the unlabeled information. Finally, experimental results on text documents demonstrate our proposed method benefits for improving the classification performance.

1 Introduction

Text classification (or categorization) is one of key problems in text mining. It aims to automatically assign unlabeled documents to one or more predefined classes based on their content. A number of statistical and machine learning techniques have been developed to solve the problem of text classification, e. g., regression model, k-nearest neighbor, decision tree, Naïve Bayes, Support Vector Machines, etc. [4,9]. There are several difficulties for this task. First, the existence of polysemy (a word contains multiple meanings) and synonym (different words express the same concepts) makes it hard to form appropriate classification models [3,11,8]. Second, documents usually representing by "bag of words" are inherently in very high dimension feature space. It increases the chance of overfitting [9]. Third, due to the difficult and tedious nature of labeling, training samples are sometimes extremely limited, this makes it difficult to make decisions with high confidence [11].

In order to solve the problems of polysemy and synonym, researchers usually adopt the Latent Semantic Indexing (LSI) technique [3] due to its empirically effective at overcoming these problems in text classification. LSI also succeeds in a wide variety of learning tasks, such as search and retrieval [3], classification [11,8] and information filtering [5,6], etc. It is a vector space approach for modeling documents, and many papers have claimed that this technique brings

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 53–60, 2009.

out the latent semantics in a collection of documents [3]. Typically, LSI is applied in an unsupervised paradigm. It is based on the well known mathematical technique called Singular Value Decomposition (SVD) and maps the original term-document matrix into a low dimensional space. Recently, researchers have incorporated class information into LSI for text classification. For example, [10] has presented a technique called Supervised LSI which is based on iteratively identifying discriminative eigenvectors from class-specific LSI representations. [1] has introduced a very simple method, called "sprinkling" to integrate class information into LSI. The basic idea of [1] is to construct an augmented term-document matrix by encoding class labels as artificial terms and appending to training documents. LSI is performed on the augmented term-document matrix, where class-specific word associations being strengthened. These pull documents and words belonging to the same class closer to each other in a indirect way. [2] extends the idea of [1] by adaptively changing the length of sprinkling terms based on results from confusion matrices.

One problem is that the above supervised LSI techniques do not take into account useful background knowledge from unlabeled text documents. This reduces a chance to improve the performance of LSI in text categorization. After literature review, we found that [11] has incorporated background knowledge into LSI to aid classifying text categories. However, the LSI-based method in [11] does not consider the class label information, which is the most important information, in the text classification.

Hence, we aim at the above mentioned problems and utilize the advantages of LSI to improve the performance of classifying text documents. More specifically, we incorporate the sprinkling terms, which can capture relationships between classes, into original LSI and expand the term-by-document matrix by using the background knowledge from unlabeled text documents. We then validate the proposed method through detailed experiments on three text datasets from the CMU text mining group. The results show that our method can truly improve classification accuracy.

The rest of this paper is organized as followings: In Section 2, we give a brief introduction to the LSI-based techniques. In Section 3, we detail the procedure of the proposed new sprinkled LSI with background knowledge method. In Section 4, we present the experimental setup and results. Finally, the paper is concluded in Section 5.

2 Related Works

2.1 Latent Semantic Indexing

Latent Semantic Indexing [3] is based on the assumption that there is an underlying semantic structure in textual data, and that the relationship between terms and documents can be re-described in this semantic structure form. Textual documents are usually represented as vectors in a vector space. Each position in a vector corresponds to a term. If the term does not appear in the document, the value for the corresponding position equal to 0 and it is positive otherwise.

Hence, a corpus is deemed into a large term-by-document ($t \times d$) matrix X, where, x_{ij} corresponds to the presence or absence of a term (in row i) in a document (in column j), $X_{\cdot j}$ represents the vector of a document j, $X_{i\cdot}$ represents the vector of a word i. The value of x_{ij} can have other expressions, e.g., term frequency (tf), term frequency inverse document frequency (tf-idf), etc.

This matrix, X, is typically very sparse, as most documents consist of only a small percentage of the total number of terms occurred in the full corpus of documents. Due to the nature of text documents, where a word can have ambiguous meanings and each concept can be represented by many different words, LSI reduces the large space to a small space hoping to capture the true relationships between documents by Singular Value Decomposition (SVD), i.e., $X = TSD^{\top}$, where $^{\top}$ is the transpose of a matrix, T and D are orthogonal matrices and S is the diagonal matrix of singular values and the diagonal elements of S are ordered by magnitude. To reduce the dimension, the smallest k values in S can simplified be set to zero. The columns of T and D that correspond to the values of S that were set to zero are deleted. The new product of these simplified three matrices consists of a matrix $\tilde{X}$ that is an approximation of the term-by-document matrix, $\tilde{X} = T_k S_k D_k^{\top}$.

These factors can be deemed as combining meanings of different terms and documents; and documents can be reexpressed using these factors. When LSI is used for retrieval, a query, q, is represented in the same new small space that the document collection is represented in, forming a pseudo-document. This is done by multiplying the transpose of the term vector of the query with matrices T_k and S_k^{-1} [3], $X_q = q^{\top} T_k S_k^{-1}$.

Once the query is represented in this way, the distance between the query and documents can be computed using the cosine metric (or Euclidean distance), which measures similarity between documents. LSI returns the distance between the query and all documents in the collection. Those documents that have higher cosine value (or smaller Euclidean distance) than a given threshold can be returned as relevant to the query. For text classification, k-nearest neighbor (kNN) or Support Vector Machine (SVM) are then applied on the reduced features to get the decision rule or boundary.

2.2 Sprinkling

In [1], a sprinkling LSI was developed by augmenting artificial terms based on class labels in the term-by-document matrix. As the usual manner of LSI, SVD is then performed on the augmented term-by-document matrix. Noisy dimensions corresponding to small singular values are deleted and a low rank approximated matrix is constructed. When in the test phase, in order to make the training document representations compatible with test documents, the sprinkled dimensions are removed. Standard classification methods, e.g., kNN and SVM are then applied in the new represented training features. The inherent reason for this method is that the sprinkled term can add contribution to combine the class information of text documents in the classification procedure.

Further, in [2], an adaptive sprinkling LSI is proposed to adjust the number of sprinkled terms based on the complexity of the class decision boundary and the classifier's classification ability. The number of sprinkled terms for each class is then determined by the confusion matrices: when it is hard to distinguish two classes, it induces more errors in the confusion matrix. In this case, more sprinkling terms are introduced for those classes.

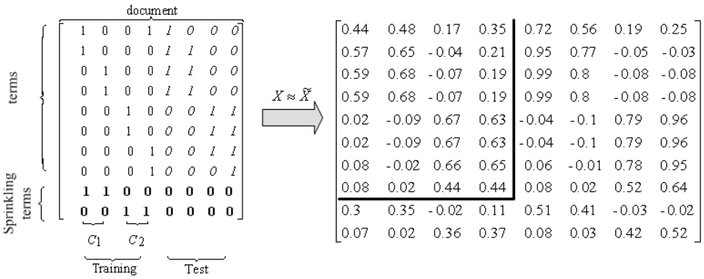
3 Our Approach

In this section, we detail our proposed method on how to incorporate the class information and unlabeled information from background knowledge into the LSI technique. Fig. 1 gives a demonstration.

Assume that there are two classes documents, each consisting of two documents and two other unlabeled documents. We then extend the term-by-document matrix, which is represented by term frequency, by adding sprinkling terms, where we set 1 when the documents are from the same class, zero when they are not in the same class or no label information is given. We then perform the SVD method and obtain an approximation of the term-by-document matrix, X_k, which is reconstructed by selecting the two largest values in S, as shown in Fig. 1. It is easy to see that the approximate matrix has reconstructed the value distribution of data.

Hence, we summarize the procedure as follows:

1. Construct a term frequency matrix, $X \in \mathbb{R}^{f \times (N_L + N_U)}$, by using labeled and unlabeled data, where f is the number of words, N_L and N_U corresponds to the number of labeled and unlabeled documents, respectively;
2. Expand X to a sprinkled matrix, $X_e \in \mathbb{R}^{(f+d) \times (N_L + N_U)}$, by adding sprinkling terms with ones for the labeled data and terms with zeros for the unlabeled data, where d is the length of sprinkling terms;
3. Perform Singular Value Decomposition (SVD) on the expanded matrix, i.e., $X_e = TSD^\top$, where $T \in \mathbb{R}^{(f+d) \times (f+d)}$ is an orthogonal matrix, $S \in \mathbb{R}^{(f+d) \times (N_L + N_U)}$ is a diagonal matrix, and $D \in \mathbb{R}^{(N_L + N_U) \times (N_L + N_U)}$ is another orthogonal matrix;



$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \xrightarrow{X \approx \tilde{X}} \begin{bmatrix} 0.44 & 0.48 & 0.17 & 0.35 & 0.72 & 0.56 & 0.19 & 0.25 \\ 0.57 & 0.65 & -0.04 & 0.21 & 0.95 & 0.77 & -0.05 & -0.03 \\ 0.59 & 0.68 & -0.07 & 0.19 & 0.99 & 0.8 & -0.08 & -0.08 \\ 0.59 & 0.68 & -0.07 & 0.19 & 0.99 & 0.8 & -0.08 & -0.08 \\ 0.02 & -0.09 & 0.67 & 0.63 & -0.04 & -0.1 & 0.79 & 0.96 \\ 0.02 & -0.09 & 0.67 & 0.63 & -0.04 & -0.1 & 0.79 & 0.96 \\ 0.08 & -0.02 & 0.66 & 0.65 & 0.06 & -0.01 & 0.78 & 0.95 \\ 0.08 & 0.02 & 0.44 & 0.44 & 0.08 & 0.02 & 0.52 & 0.64 \\ 0.3 & 0.35 & -0.02 & 0.11 & 0.51 & 0.41 & -0.03 & -0.02 \\ 0.07 & 0.02 & 0.36 & 0.37 & 0.08 & 0.03 & 0.42 & 0.52 \end{bmatrix}$$

Fig. 1. An example of the term-by-document matrix transforming by Sprinkling LSI with background knowledge

4. Determine columns of T and D corresponding to the k largest eigen values in S and discard the additional sprinkling terms. Hence, finally, $T_k \in \mathbb{R}^{f \times k}$, $S_k \in \mathbb{R}^{k \times k}$, a diagonal matrix, and $D_k \in \mathbb{R}^{k \times N_L}$;
5. Transform training data to new features, D_k, and for a test data, $q \in \mathbb{R}^f$, the corresponding new feature is calculated by $\tilde{q} = q^\top T_k S_k^{-1} \in \mathbb{R}^k$. The length of new features becomes shorter since $k \ll f$;
6. Perform classification algorithms on transformed new features.

4 Experiments

4.1 Experimental Setup

In this section, we evaluate the standard LSI, sprinkled LSI (SLSI), LSI with background knowledge (LSI-bg) and our proposed sprinkled LSI with background knowledge (SLSI-bg) on three datasets: the WebKB dataset, the 20 newsgroups dataset and the industry sector (IS) dataset, from CMU text mining group[1].

Test is performed in the transductive mode: test data are considered as unlabeled data and used for the test. Accuracy [2] is used to evaluate the performance of text classification on kNN with two different metrics, the Euclidean distance (kNNE) and the cosine similarity (kNNC).

Before performing classification, text processing, skipping headers, skipping html, etc., is run by the rainbow package[2]. The option of whether or not using stoplist is detailed in the following data sets description. Further, we remove the words that only occur once and choose the top 1000 with highest mutual information gain [2].

4.2 Data Sets

Three standard text documents data sets are tested in the experiments:

20 newsgroups dataset: This data set is a collection of 20,000 UseNet news postings into 20 different newsgroups [7] including seven sub-trees: alt, comp, rec, sci, soc, talk and misc. We use rec sub-tree which consists of 4 classes in the experiment. 500 documents from each class are selected [2]. Words in the standard SMART stoplist is removed from the dataset. Finally, this dataset forms a 1000 × 2000 term-by-document matrix.

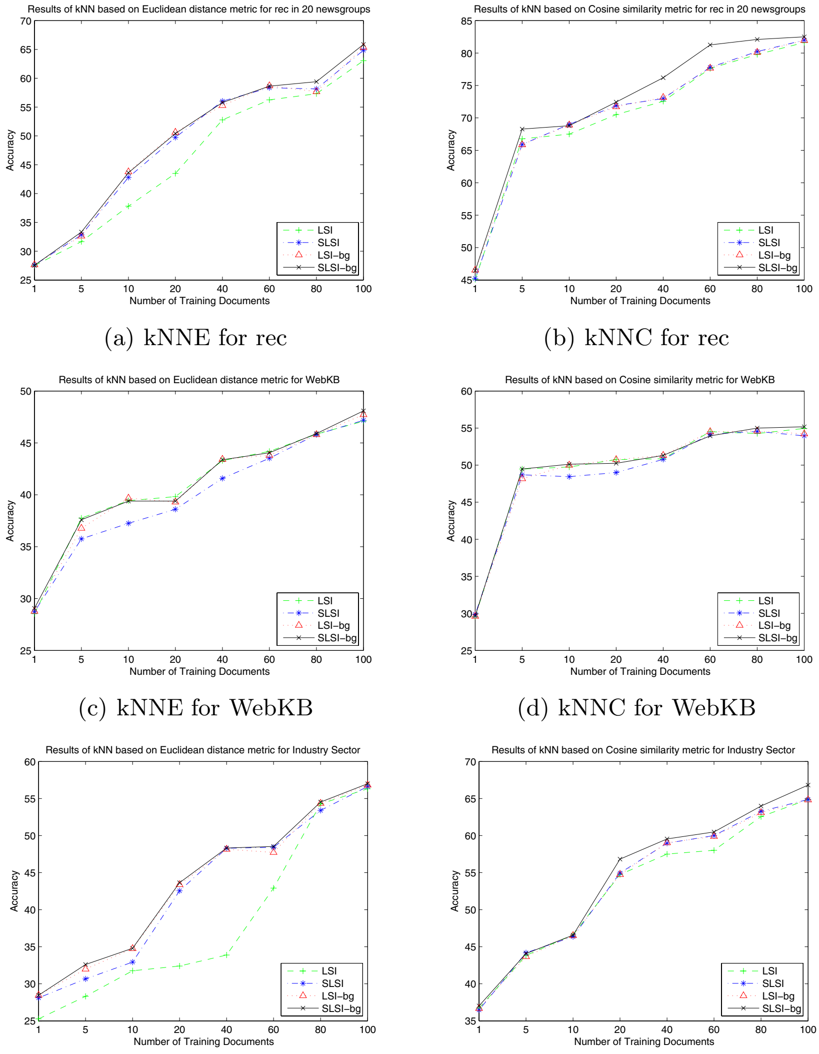
WebKB dataset: It is the 4 universities data set from the World Wide Knowledge Base (WebKb) project of the CMU text learning group [3]. There are 8,282 pages were manually classified into the following seven categories: student, faculty, staff, department, course, project and other. Here, again, each class consists of 500 documents. Hence, the categories of staff and department do not use in the experiment. Stoplist option is not used in this dataset. The dataset forms a 1000 × 2500 term-by-document matrix.

[1] http://www.cs.cmu.edu/~TextLearning/datasets.html

[2] http://www.cs.umass.edu/~mccallum/bow/rainbow/

[3] http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/

Industry sector dataset: This dataset consists of 7 industry sector information: basic materials, energy, finance, healthcare, technology, transportation and utilities. Here, again, we choose 500 documents for each class. Hence, only categories of basic materials, finance, technology and transportation are considered in the experiment. Stoplist option is used in this dataset. Finally, the dataset consists of a 1000 × 2000 term-document matrix.



(a) kNNE for rec

(b) kNNC for rec

(c) kNNE for WebKB

(d) kNNC for WebKB

(e) kNNE for IS

(f) kNNC for IS

Fig. 2. Results of kNN based on two different distance metrics for rec, WebKB, and IS datasets

4.3 Experimental Results

In the experiments, we let the number of labeled data be 1, 5, 10, 20, 40, 60, 80 and 100 to test the effect of the number of training data. In classification using kNN method, features generated by LSI, SLSI, LSI-bg and SLSI-bg are used.

The length of sprinkling term for each class is set to 4 as [1]. In the kNN, k is set to 1. Results are performed on 10 runs to get the average accuracies.

Fig. 2 presents the results of kNNE and kNNC on the rec sub-tree of 20 newsgroups, the WebKB, and the industry sector datasets, respectively. From the results, we have the following observations:

- As the number of labeled training samples increases, the accuracies for all four features, LSI, SLSI, LSI-bg, SLSI-bg increase correspondingly. SLSI-bg is the best overall for all three datasets.
- The performance usually has a large increase when the number of training sample increase from 1 to 5, especially using the kNNC method. For example, when using the kNNC method, it is over 41% improvement on the rec subtree and over 70% improvement on other two datasets.
- The performance of kNNC outperforms that of kNNE greatly for all three datasets. Especially, the improvement using the same feature is obviously significant, at least 20% improvement, for the rec subtree in 20 newsgroup dataset. Even for the WebKB dataset, although the improvement is not large when the number of labeled data is 1, the performance increases significantly when the number of labeled data is larger than 1.
- For the industry sector dataset, it is shown that LSI is rather worse in this dataset under the kNNE metric, especially when the number of training samples equals 20, 40, 60; while other three features can get relative better results. This means that other features can help for the classification procedure.

5 Conclusions

In this paper, we consider two problems: the existence of poylsemy and synonym, and the huge useful unlabeled text documents, in text classification. We propose a novel Latent Semantic Indexing expression which utilizes the class relationship and background knowledge embedding in text documents. We perform detailed experimental comparisons to test the effects of class relationship and background knowledge. Experimental results demonstrate our proposed method is promising in text classification.

There are still several works need to be considered. For example, why and how this combination will help increase the performance of text classification. Developing theoretical framework for analyzing the difference between our method and original LSI and providing theoretical guidance are significant works.

Acknowledgments

The work described in this paper was fully supported by the grants from the Research Grants Council of the Hong Kong SAR, China (Project No. CUHK 4125/07E).

References

1. Chakraborti, S., Lothian, R., Wiratunga, N., Watt, S.: Sprinkling: Supervised latent semantic indexing. In: Lalmas, M., MacFarlane, A., Rüger, S.M., Tombros, A., Tsikrika, T., Yavlinsky, A. (eds.) ECIR 2006. LNCS, vol. 3936, pp. 510–514. Springer, Heidelberg (2006)
2. Chakraborti, S., Mukras, R., Lothian, R., Wiratunga, N., Watt, S., Harper, D.: Supervised latent semantic indexing using adaptive sprinkling. In: Proceedings of IJCAI 2007, pp. 1582–1587 (2007)
3. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. Journal of the American Society of Information Science 41(6), 391–407 (1990)
4. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. John Wiley & Sons, Chichester (2000)
5. Dumais, S.T.: Using lsi for information filtering: Trec-3 experiments (1995)
6. Gee, K.R.: Using latent semantic indexing to filter spam. In: Proceedings of the 2003 ACM symposium on Applied computing, pp. 460–464 (2003)
7. Lang, K.: NewsWeeder: learning to filter netnews. In: Proceedings of ICML 1995, pp. 331–339. Morgan Kaufmann publishers Inc., San Mateo (1995)
8. Liu, T., Chen, Z., Zhang, B., Ma, W.y., Wu, G.: Improving text classification using local latent semantic indexing. In: Proceedings of ICDM 2004, pp. 162–169 (2004)
9. Sebastiani, F.: Machine learning in automated text categorization. ACM Computing Surveys 34(1), 1–47 (2002)
10. Sun, J.-T., Chen, Z., Zeng, H.-J., Lu, Y.-C., Shi, C.-Y., Ma, W.-Y.: Supervised latent semantic indexing for document categorization. In: Perner, P. (ed.) ICDM 2004. LNCS, vol. 3275, pp. 535–538. Springer, Heidelberg (2004)
11. Zelikovitz, S., Hirsh, H.: Using LSI for text classification in the presence of background text. In: Paques, H., Liu, L., Grossman, D. (eds.) Proceedings of CIKM 2001, pp. 113–118 (2001)

Comparison of Cluster Algorithms for the Analysis of Text Data Using Kolmogorov Complexity

Tina Geweniger[1], Frank-Michael Schleif[1], Alexander Hasenfuss[2], Barbara Hammer[2], and Thomas Villmann[1]

[1] University Leipzig, Dept. of Medicine, 04103 Leipzig, Germany
{schleif,villmann}@informatik.uni-leipzig.de, tg@sonowin.de
[2] Clausthal Univ. of Tech., Inst. of CS, 38678 Clausthal, Germany
{hasenfuss,hammer}@inf.tu-clausthal.de

Abstract. In this paper we present a comparison of multiple cluster algorithms and their suitability for clustering text data. The clustering is based on similarities only, employing the Kolmogorov complexity as a similiarity measure. This motivates the set of considered clustering algorithms which take into account the similarity between objects exclusively. Compared cluster algorithms are Median kMeans, Median Neural Gas, Relational Neural Gas, Spectral Clustering and Affinity Propagation.

Keywords: Cluster algorithm, similarity data, neural gas, spectral clustering, message passing, kMeans, Kolmogorov complexity.

1 Introduction

In the last years a variety of vector based clustering methods like self-organizing maps [7], neural gas (NG) [9] and affinity propagation (AP) [6] have been developed for a wide range of areas, e.g. bioinformatics, business, and robotics. Recent developments focus on algorithms which are purely based on the analysis of similarities between data. Beside AP respective algorithms are Median k-Means, Median and Relational NG [3,1] or spectral clustering [8] for example. These approaches have in common that they relax the condition that the objects of interest have to be embedded in a metric space. Instead, the only information used in these algorithms are the pairwise similarities. This ability provides greater flexibility of the algorithm if an embedding of data is impossible but a similarity description is available, for instance by external expert rating.

In this paper we compare the above mentioned algorithms in two experimental settings of text clustering. We consider two types of text sources: the first data are taken from the Multilingual Thesaurus of the European Union Eurovoc[1]. This database consists of laws and regulations in different categories whereas each text is available in multiple languages. It can be expected that the data

[1] Obtained from: http://europa.eu/eurovoc/sg/sga_doc/eurovoc_dif!SERVEUR/menu!prod!MENU?langue=EN

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 61–69, 2009.

should be clustered according to language specific features on the one hand side. On the other hand, contents specific features should also provide structural information for clustering. The second data set is a series of psychotherapy session transcripts of a complete psychodynamic psychotherapy with a respective clinical assessment of the session [10]. Here, it is known that narrative constructs provide information about the therapy state and, hence, should also be appearing in cluster structures. Both data sets are clustered by means of the priorly given methods. The results are compared in terms of cluster agreement and the relation to the external description. As a measure for the cluster agreement we employ the Cohens-Kappa and variants.

The paper is organized as follows: first the encoding of the data is explained. After a short overview of the algorithms, the data sets are briefly described and finally the experimental results are presented and discussed.

2 Theoretical Background

Appropriate embedding of text data into metric spaces is a difficult task [7]. An approach is the analysis of textural data based on their Kolmogorov-Complexity [2]. It is based on the minimal description length (MDL) Z_x of a single document x and pairwise combined documents $Z_{x,y}$. The respective normalized information distance is given by:

$$NID_{x,y} = \frac{Z_{xy} - min(Z_x, Z_y)}{max(Z_x, Z_y)} \tag{1}$$

The normalized information distance is a similarity measure (distance metric) [2]. In particular, it is positive definite, i.e. $NID_{x,y} \geq 0$ with $NID_{x,x} = 0$, and symmetric $NID_{x,y} = NID_{y,x}$. Usually, the MDL is estimated by the compression length z according to a given standard compression scheme or algorithm. Then the NID is denoted as normalized compression distance (NCD) [2]. Due to technical reasons $NCD_{x,y}$ is non-vanishing in general but takes very small values [2]. Further, it violates usually the symmetry property in the sense that $NCD_{x,y} - NCD_{y,x} = \delta$ with $0 < \delta \ll 1$. Therefore, the symmetrized variant is frequently applied $NCD^s_{x,y} = \frac{(NCD_{x,y} + NCD_{y,x})}{2}$.

To estimate NCD, we used the Lempel-Ziv-Markow-Algorithm (LZMA) provided by the 7-Zip file archiver for the compression of the text data. z_x and z_y are the lengths of the compressed data sets T_x and T_y respectively. To obtain z_{xy} the two texts T_x and T_y are first concatenated to T_{xy} and subsequently compressed. In this way for all data pairs (x^i, x^j) the similarity (distance) $d_{ij} = NCD_{x^i,x^j}$ is calculated in both data sets (separately).

3 Algorithms

The role of clustering is to decompose a given data set $X = \{x_1, \ldots, x_n\}$ into clusters $C_1, \ldots, C_k \subset X$ such that the clusters are as homogeneous as possible.

For crisp clustering as considered in the following, the sets C_i are mutually disjoint, and they cover the set X. We assume that data are characterized by pairwise dissimilarities d_{ij} specifying the dissimilarity of the data points x_i and x_j. For euclidean data, d_{ij} could be given by the squared euclidean metric, however, in general, every symmetric square matrix is appropriate. However, we assume here that only the distances are known but not the data objects itself. This restricts the set of applicable cluster algorithms. Following, we briefly describe recently developed approaches as well as classical ones which will later be compared.

3.1 Median k-Means

Median k-means (MKM) is a variant of classic *k-means* for discrete settings. The cost function for MKM is given by

$$E = \sum_{i=1}^{n} \sum_{j=1}^{p} X_{I(x^j)}(i) \cdot d\left(x^j, w^i\right) \tag{2}$$

where n is the cardinality of the set $W = \left\{w^k\right\}$ of the prototypes and p the number of data points. $X_{I(x^j)}(i)$ is the characteristic function of the winner index $I(x^j)$, which refers to the index of the prototype with minimum distance to x^j (winner).

E is optimized by iteration through the following two adaptation steps until convergence is reached.

1. determine the winner $I(x^j)$ for each data point x^j
2. Since for proximity data only the distance matrix is available the new prototype i has to be chosen from the set X of data points with $w^i = x^l$ where

$$l = \underset{l'}{\operatorname{argmin}} \sum_{j=1}^{p} X_{I(x^j)}(l) \cdot d(x^j, x^{l'}) \tag{3}$$

3.2 Median Neural Gas

A generalization of MKM incorporating neighborhood cooperativeness for faster convergence and better stability and performance is the Median Neural Gas (MNG). The respective cost function is

$$E_{MNG} = \sum_{i=1}^{n} \sum_{j=1}^{p} h_\lambda(k_i(x^j, W)) \cdot d(x^j, w^i) \tag{4}$$

with $h_\lambda(k_i(x^j, W))$ being the Gaussian shaped neighborhood function $h_\lambda(t) = exp(-t/\lambda)$ ($\lambda < 0$) and

$$k_i(x^j, W) = \# \left\{w^l | d(x^j, w^l) < d(x^j, w^i)\right\} \tag{5}$$

the winning rank. Then E_{MNG} can be optimized by iterating the following procedure:

1. $k_{ij} = k_i(x^j, W)$
2. and assuming fixed k_{ij} the prototype i is chosen as the data point with $w^i = x^l$ where

$$l = \underset{l'}{\text{argmin}} \sum_{j=1}^{p} h_\lambda(k_{ij}) \cdot d(x^j, x^{l'}).$$

3.3 Relational Neural Gas

Relational neural gas as proposed in [1] is based on a similar principle as MNG, whereby prototype locations can be chosen in a more general way than in MNG. Standard batch neural gas [3] has been defined in the euclidean setting, i.e. $x^i \in \mathbb{R}^m$ for some m. It optimizes the cost function $\frac{1}{2}\sum_{ij} \exp(-k_{ij}/\sigma^2)\|x^i - w^j\|^2$ with respect to the prototypes w_j where k_{ij} as above but using the Euclidean distance. $\sigma > 0$ denotes the neighborhood cooperation. For vanishing neighborhood $\sigma \to 0$, the standard quantization error is obtained. This cost function can be optimized in batch mode by subsequent optimization of prototype locations and assignments. Unlike k-means, neighborhood cooperation yields a very robust and initialization insensitive behavior of the algorithm.

The main observation of relational clustering is that optimum prototypes fulfill the relation $w^j = \sum_i \alpha_{ji} x^i$ with $\sum_i \alpha_{ji} = 1$. Therefore, the distance $\|x^i - w^j\|^2$ can be expressed solely in terms of the parameters α_{ji} and the pairwise distances $D = (d^2_{ij})_{ij}$ of the data as

$$\|x^i - w^j\|^2 = (D \cdot \alpha_j)_i - 1/2 \cdot \alpha_j^t \cdot D \cdot \alpha_j. \qquad (6)$$

Therefore, it is possible to find a formulation of batch NG which does not rely on the explicit embedding of data in a vector space:

init α_{ji} with $\sum_i \alpha_{ji} = 1$
repeat

- compute the distance $\|x^j - w^i\|^2$
- compute optimum assignments k_{ij} based on this distance matrix
- compute parameters $\tilde{\alpha}_{ij} = \exp(-k_{ij}/\sigma^2)$
- normalize $\alpha_{ij} = \tilde{\alpha}_{ij} / \sum_j \tilde{\alpha}_{ij}$

Obviously, this procedure can be applied to every symmetric dissimilarity matrix D, resulting in relational neural gas (RNG). The algorithm can be related to the dual cost function of NG:

$$\sum_i \frac{\sum_{ll'} \exp(-k_{il}/\sigma^2) \cdot \exp(-k_{il'}/\sigma^2) \cdot d_{ll'}}{4 \sum_l \exp(-k_{il}/\sigma^2))}$$

Since prototypes of RNG are represented virtually in terms of weighting factors α_{ij}, the algorithm yields a clustering rather than a compact description of the classes in terms of prototypes. However, it is possible to approximate the clusters by substituting the virtual prototypes w^j by its respective closest exemplar x_i in the data set X. We refer to this setting as 1-approximation of RNG.

3.4 Spectral Clustering

Spectral clustering (SC) offers a popular clustering method which is based on a graph cut approach, see e.g. [8]. The idea is to decompose the vertices of the graph into clusters such that the resulting clusters are as close to connected components of the graph as possible. More precisely, assume vertices are enumerated by $1, \ldots, n$ corresponding to the data points x_i and undirected edges $i - j$ weighted with p_{ij} indicate the similarity of the vertices. We choose $p_{ij} = -d_{ij} + \min_{ij} d_{ij}$, but alternative choices are possible, as described in [8]. Denote the resulting matrix by P, D denotes the diagonal matrix with vertex degrees $d_i = \sum_i p_{ij}$. Then normalized spectral clustering computes the smallest k eigenvectors of the normalized graph Laplacian $D^{-1} \cdot (D - P)$. The components of these eigenvectors constitute n data points in $\mathbb{R}^k$ which are clustered into k classes using a simple algorithm such as k-means. The index assignment gives the clusters C_i of X.

The method is exact if the graph decomposes into k connected components. As explained in [8], it constitutes a reasonable approximation to the normalized cut optimization problem $\frac{1}{2} \sum_i W(C_i, C_i^c)/\text{vol}(C_i)$ for general graphs, where $W(A, A^c) = \sum_{i \in A, j \notin A} p_{ij}$ denotes the weights intersected by a cluster A and $\text{vol}(A) = \sum_{j \in A} d_j$ the volume of a cluster A. Further, for normalized SC, some form of consistency of the method can be proven [8].

3.5 Affinity Propagation

Affinity propagation (AP) constitutes an exemplar-based clustering approach as proposed in [6]. Given data points and pairwise dissimilarities d_{ij}, the goal is to find k exemplars x_i such that the following holds: if data points x^i are assigned to their respective closest exemplar by means of $I(i)$, the overall quantization error $\frac{1}{2} \sum_i d_{i,I(i)}$ should be minimum. This problem can be alternatively stated as finding an assignment function $I : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that the costs $-\frac{1}{2} \sum_i d_{i,I(i)} + \sum_i \delta_i(I)$ are maximum. Thereby, $\delta_i(I) = \begin{cases} -\infty & \text{if } I(i) \neq i, \exists j I(j) = i \\ 0 & \text{otherwise} \end{cases}$ punishes assignments which are invalid, because exemplar i is not available as a prototype but demanded as exemplar by some point j. Note that, this way, the number of clusters is not given priorly but it is automatically determined by the overall cost function due to the size of self-similarities $-d_{ii}$. These are often chosen identical for all i and as median value or half the average similarities. Large values $-d_{ii}$ lead to many clusters whereas small values lead to only a few or one cluster. By adjusting the diagonal d_{ii}, any number of clusters in $\{1, \ldots, n\}$ can be reached. AP optimizes this cost function by means of a popular and efficient heuristics. The cost function is interpreted as a factor graph with discrete variables $I(i)$ and function nodes $\delta_i(I)$ and $d_{i,I(i)}$. A solution is found by the max-sum-algorithm in this factor graph which can be implemented in linear time based on the number of dissimilarities d_{ij}. Note that $-d_{ij}$ can be related to the log probability of data point i to choose exemplar j, and $\delta_i(I)$ is the log probability of a valid assignment for i as being an exemplar or not. Thus, the max-sum algorithm can be interpreted as an approximation to compute assignments with maximum probability in the log-domain.

While spectral clustering yields decompositions of data points into clusters, affinity propagation also provides a representative exemplar for every cluster. This way, the clustering is restricted to specific types which are particularly intuitive. Further, unlike spectral clustering, the number of clusters is specified only implicitly by means of self-similarities.

4 Experiments and Results

4.1 Measures for Comparing Results

To measure the agreement of two different cluster solutions we applied Cohen's Kappa κ_C [4]. Fleiss' Kappa κ_F as an extension of Cohen's Kappa is suitable for measuring the agreement of more than two classifiers [5]. For both measures yields the statement that if they are greater than zero the cluster agreements are not random but systematic. The maximum value of one is perfect agreement.

4.2 'Eurovoc' Documents

The first data set consists of a selection of documents from the multilingual Thesaurus of the European Union "'Eurovoc"'. This thesaurus contains thousands of documents which are available in up to 21 languages each. For the experiments we selected a set of 600 transcripts in 6 different languages - 100 transcripts with the same contents in English, German, French, Spanish, Finnish and Dutch respectively. These transcripts can roughly be sorted into 6 different categories: International Affairs, Social Protection, Environment, Social Questions, Education and Communications, and Employment and Working Conditions.

First the distances between data are calculated according to $d_{ij} = NCD_{x^i,x^j}$for the whole data set giving a large 600 × 600 matrix and, for each language set separately, yielding 6 small 100 × 100 matrices.

The first calculations in this section are based on the 600 × 600 matrix: As a first step the complete set containing all 600 documents was clustered into *six* groups using the above mentioned algorithms to investigate whether the language structure influences the clustering. It can be observed that all cluster solutions of the different methods are identical and exactly separating the data set according to the languages. In the second step we initialized a clustering into 12 clusters to examine the content based information for clustering. It can be observed that again a clean separation regarding to the languages was achieved. Yet, the segmentation within a single language was more or less irregular. For some languages there was no further break down at all, while others were separated into up to four different clusters. Hence, it seems that the language specifics dominate the clustering. Thereby, this behavior was shown more or less by all cluster algorithms. This fact is emphasized by the similarity of the cluster solutions judged in terms of Fleiss' Kappa (overall agreement) $\kappa_F = 0.6072$ referred as a substantial agreement. The the agreements of every two algorithms are estimated by Cohen's Kappa κ_C which also show a clear match:

	MNG	RNG	SC	AP
k-Means	0.56	0.50	0.59	0.73
MNG	—	0.58	0.54	0.73
RNG	—	—	0.57	0.63
SC	—	—	—	0.66

Noticeable is the clustering obtained by AP. Each language is separated into two clusters, which are almost identical with respect to the language sets. Measuring the similarity of the clusters between the different languages gives $\kappa_F = 0.8879$, a perfect agreement.

In the next step we examined each language separately using the 100 × 100 matrices. According to the given 6 text categories, we performed each clustering into 6 clusters. At first, we have to mention that the resulted cluster solutions do not reflect the category system. This is observed for all languages and all algorithms. However, within each language the behavior of the different cluster approaches is more or less similar, i.e. comparing the cluster solutions gives high Kappa values above 0.4123 (moderate agreement). As an example, for English the solutions are depicted in Fig.1a). with Fleiss' Kappa $\kappa_F =$ 0.5324 (moderate agreement). However, the averaged performance of the several

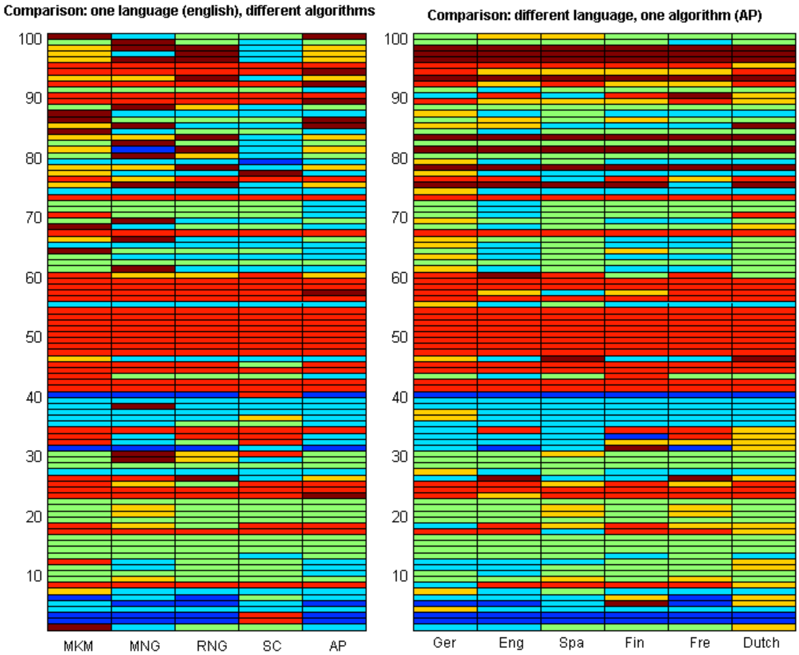


Fig. 1. a) (left) Comparison of the cluster solutions (six clusters) for the different algorithms. A moderate agreement can be observed. b) (right) Cluster solutions for the different languages obtained by AP-clustering. The similarity of the cluster results is high.

algorithms according to the different languages varies. Despite AP and RNG, all algorithms show an instable behavior. AP and RNG offer similar cluster assignments independent from the language giving $\kappa_F = 0.5766$ and $\kappa_F = 0.3454$, respectively (see Fig.1b). This leads to the conclusion that the contents of the text can be clustered adequately in each language.

4.3 Psychotherapy Transcripts

The second data set was a set of text transcripts of a series of 37 psychotherapy session dialogs of a psychodynamic therapy. Clustering these texts, using the NCD-distance as above, was again accomplished by applying all algorithms, here preferring a two-cluster solution according to the fact that the therapy was a two-phase process with the culminating point around session 17 [10]. The latter fact is based on the evaluation of several clinical therapy measures [10]. Except SC, all algorithms cluster the data in a similar way such that the two process phases are assigned to separate clusters ($\kappa_F = 0.77$). This coincides with the hypothesis that narratives of the psychotherapy can be related to the therapeutic process.

4.4 Conclusions

In this paper we investigated the behavior of different cluster algorithms for text clustering. Thereby we restricted ourself to such algorithms, which only take the distances between data into account but not the objects to be clustered itself. As distance measure we used the information distance. It can be concluded that, if texts from different languages are available (here 'Eurovoc'-documents), this language structure dominates the clustering, independent from the cluster algorithm. An overall moderate agreement between the different approaches is observed. Content specific clustering (separated in each language) is more difficult. The overall agreement is good as well but with instable results for the different languages depending on the approaches. Here, AP and RNG (with curtailments) show the most reliable results. The content specific discrimination ability is also verified for a text data base of psychotherapy session transcripts, which can be related to different therapy phases. This phase structure is nicely verified by the text clustering by almost all cluster algorithms.

Acknowledgement. The authors would like to thank Matthias Ongyerth for his contributions in early experiments on AP and SC.

References

1. Hammer, B., Hasenfuss, A.: Relational neural gas. Künstliche Intelligenz, 190–204 (2007)
2. Cilibrasi, R., Vitányi, P.: Clustering by compression. IEEE Transactions on Information Theory 51(4), 1523–1545 (2005)

3. Cottrell, M., Hammer, B., HasenfuSS, A., Villmann, T.: Batch and median neural gas. Neural Networks 19, 762–771 (2006)
4. Dou, W., Ren, Y., Wu, Q., Ruan, S., Chen, Y., Constans, D.B.A.-M.: Fuzzy kappa for the agreement measure of fuzzy classifications. Neurocomputing 70, 726–734 (2007)
5. Fleiss, J.: Statistical Methods for Rates and Proportions, 2nd edn. Wiley, New York (1981)
6. Frey, B., Dueck, D.: Clustering by message passing between data points. Science 315, 972–976 (2007)
7. Kohonen, T.: Self-Organizing Maps. Springer Series in Information Sciences, vol. 30. Springer, Heidelberg (1995) (2nd extended edn. 1997)
8. Luxburg, U.V.: A tutorial on spectral clustering. Statistics and Computing 17(4), 395–416 (2007)
9. Martinetz, T.M., Berkovich, S.G., Schulten, K.J.: 'Neural-gas' network for vector quantization and its application to time-series prediction. IEEE Trans. on Neural Networks 4(4), 558–569 (1993)
10. Villmann, T., Liebers, C., Bergmann, B., Gumz, A., Geyer, M.: Investigation of psycho-physiological interactions between patient and therapist during a psychodynamic therapy and their relation to speech in terms of entropy analysis using a neural network approach. New Ideas in Psychology 26, 309–325 (2008)

Neurocognitive Approach to Clustering of PubMed Query Results

Paweł Matykiewicz[1,2], Włodzisław Duch[1],
Paul M. Zender[3], Keith A. Crutcher[3], and John P. Pestian[2]

[1] Nicolaus Copernicus University, Torun, Poland
[2] Cincinnati Children's Hospital Medical Center, Cincinnati, Ohio, USA
[3] University of Cincinnati, Cincinnati, Ohio, USA

Abstract. Internet literature queries return a long lists of citations, ordered according to their relevance or date. Query results may also be represented using Visual Language that takes as input a small set of semantically related concepts present in the citations. First experiments with such visualization have been done using PubMed neuronal plasticity citations with manually created semantic graphs. Here neurocognitive inspirations are used to create similar semantic graphs in an automated fashion. This way a long list of citations is changed to small semantic graphs that allow semi-automated query refinement and literature based discovery.

1 Introduction

Structured electronic databases and free text information accessible via Internet completely changed the way information is searched, maintained and acquired. Most popular search engines index now well over 10 billion pages, but the quality of this information is low. Instead of boosting productivity increasingly large proportion of time is spent on searching and evaluating results. An ideal search and presentation system should be matched to the neurocognitive mechanisms responsible for understanding information [1]. Visualization of clusters of documents that are semantically related should reflect relations between configurations of neural activations in the brain of an expert. Many books have been written on various *concept mapping* or *mind mapping* techniques [2] that essentially recommend non-linear notes in form of graphs containing interrelated concepts. These techniques are also supported by a large number of software packages, known as *mind mapping* software (see the Wikipedia entry on *mind map*). There are some indications that these techniques indeed help to learn and remember written material in a better way [3]. However, creation of mind maps has so far been manual, and there is a clear need to introduce these techniques in query refinement [4] and literature based discovery [5].

We are especially interested in search and visualization of information in the life sciences domain, therefore the experiments reported below have been done on a PubMed, a collection of over 18 million citations. In [6,7] a prototype of a Visual Language (VL) system has been used on manually create semantic graphs that represent semantically related key biological concepts manually extracted from findings reported in the literature from the PubMed database. The purpose of this paper is to show that similar results can be obtained using computational methods.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 70–79, 2009.

There are several ways in which information retrieval can be improved. A novel approach introduced recently is based on asking the user a minimum number of relevant questions to disambiguate the topic in a precise way [8]. An interesting research direction is based on knowledge-based clustering techniques [9]. In this paper neurocognitive inspirations [1] are used to create semantic graphs that represent PubMed search results. In the next section the outline of this approach is presented, followed by the description of methods and experiments on a restricted PubMed domain. The neurocognitive approach increases quality of query result clusters enabling to create useful input for the Visual Language system that changes textual representation to icon-based graphical representation. The end-user is presented iconic graphs instead of a long list of citations enabling faster query refinement and literature based discovery.

2 Neurocognitive Inspirations in Information Retrieval

The purpose of the **VL project** is to test and report the effectiveness of the icon-based visualization in comparison with an existing text-based display approach to internet or database queries. The goal is a display that will enable scientists to identify biological concepts and their relationships more quickly, leading to insight and discovery. In order to make the project precise yet extensible the initial prototype is based on the Unified Medical Language System (UMLS) [10], a well-developed biomedical scientific ontology. Moreover the domain for developing the VL was narrowed to neuronal plasticity in Alzheimer disease (a field of expertise of one of the authors). Hence, to make the project rather general from a design perspective, graphic design and information visualization principles will be limited to provide a syntax of visual form for systems biology. However, systematic visualization techniques for the representation of biological concepts and their relations are applicable to the visualization of concepts within any field of biology.

For the first VL project experiment papers were reviewed manually based upon a random selection of 40 citations from PubMed resulting from search of the terms *Alzheimer Disease* and the protein *ApoE*. From these papers 20 terms were extracted that express important key concepts [7]. These concepts were represented in a semantic graph shown in the left side of fig. 1 (edges of the graph represent semantic relations between concepts manually extracted from the 40 citations).

Next, graphical designers converted each conceptual object into a visual icon, adding specific modification of shapes as one means to systematically depict basic categories of things, processes and actions. These modifiers should perform functions similar to the role adjectives or adverbs serve in natural language [11]. One visual icon systems (designed by students Sean Gresens, David Kroner, Nolan Stover and Luke Woods at the University of Cincinnati) was used to change graph shown in the left fig. 1 to an icon representation shown in the right fig. 1. Visual elements are quickly associated with the desired information that is being searched for. The final step of this project is to measure accuracy and time for completing a list of cognitive tasks that are selected and sequenced to represent the workflow of a scientist conducting a search. These measures will both demonstrate whether the effects of the display are significant, and also provide feedback for future improvements for the VL prototype.

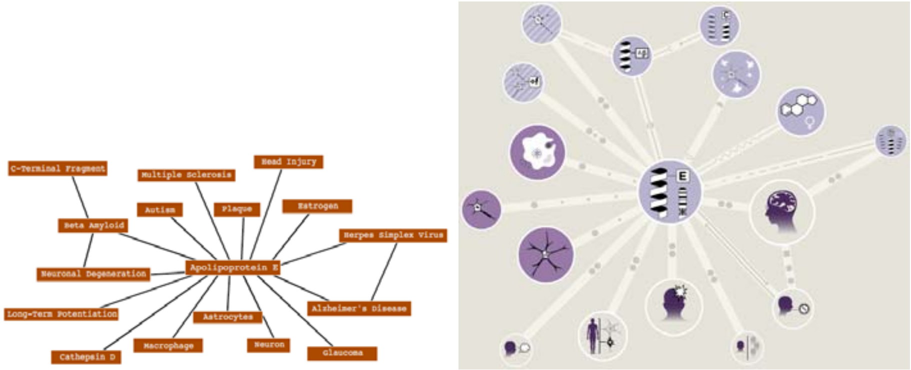


Fig. 1. Manually created textual and iconic representation of a semantic graph based on random sample of 40 publications about the Alzheimer disease and the protein ApoE

An intermediate step for the VL project is to create graphs similar to the one in the left fig. 1 in an automated fashion. This is achieved by clustering query results (similar to *Teoma, Ask, Vivisimo* or *Clusty* search engines), enhancing the initial concept space with semantically related terms and representing each cluster with by a semantic graph. Since each citation is represented by a set of biological concepts enhancing the initial set concepts with semantically related concepts is similar to automated priming effect in a human brains [12]. This means that whenever someone sees word *dog* most likely he will also think about concept *cat*. Semantic priming was studied for over 30 years but never incorporated *per se* into practical computational algorithms. Multiple evidence for priming effects comes from psychology and neuroimaging. For that reason presented here technique for information retrieval is **neurocognitively inspired** [1,13,14].

3 Methods

Large number of PubMed citations have been annotated using Medical Subject Headings (MeSH), providing keywords that characterize the content of an article. MeSH is a hierarchical controlled vocabulary created by National Library of Medicine, used also for indexing books and other documents. Moreover, MeSH is a part UMLS [10], much larger vocabulary that combines over 140 biomedical ontologies and enhance MeSH terms with additional semantic relations that come from other sources. The use of MeSH terms allows for some standardization of searches.

In our experiment a search query *"Alzheimer disease"[MeSH Terms] AND "apolipoproteins e"[MeSH Terms] AND "humans"[MeSH Terms]* was submitted to the PubMed server. It returned 2899 citations along with 1924 MeSH terms. MeSH terms are organized in 16 hierarchical trees. Concepts from only four trees were selected: *Anatomy; Diseases; Chemicals and Drugs; Analytical, Diagnostic and Therapeutic Techniques and Equipment*. This narrowed down the number of concepts to 1190. A binary document/concept matrix was created with information whether a citation had a given MeSH term assigned to it or not. Creating graphs similar to one in the left fig. 1 involves iterative use of **cluster analysis** and **automated priming** technique. Once a desired clustering quality is achieved a **semantic graphs** as an input for the VL prototype can be computed.

Cluster analysis proposed in this paper is composed of three steps: calculating a distance matrix, organizing documents into hierarchical clusters, and choosing a number of clusters based on quality measures. Since the data is binary various similarity measures from the R language statistical package called *simba* can be used [15]. Let's assume that M is a matrix with rows corresponding to documents and columns to MeSH concepts. Then $M_i^j = 1$ means that a document i has MeSH concept j assigned to it. For this experiment Legendre measure is chosen as a distance function (among fifty other measures it was found to give good results, publication in preparation), defined by [16]:

$$\mathrm{d}(\mathbf{x},\mathbf{y}) = 1 - \frac{3\langle\mathbf{x},\mathbf{y}\rangle}{\langle\mathbf{x},\mathbf{y}\rangle + \langle\mathbf{x},\mathbf{x}\rangle + \langle\mathbf{y},\mathbf{y}\rangle} \in [0,1]. \tag{1}$$

Clustering is done using Ward's minimum variance rule [17]. This method is known to produce clusters of almost equal size, but is sensitive to outliers [18]. This is not the optimal clustering algorithm but it suffice for presented here experiments (relation between clustering algorithm and automated priming will be published elsewhere).

In order to choose the optimal number of clusters a combination of four quality measures from the *clusterSim* R language statistical package is used [19]. This includes Davies-Bouldin's index which needs to be minimized [20], Calinski-Harabasz pseudo F-Statistic which needs to be maximized [21], Hubert-Levine internal cluster quality index which needs to be minimized [22] and Rousseeuw Silhouette internal cluster quality index which needs to be maximized [23]. A combination of all four indexes can be normalized between 0 and 4 so that 4 means perfect agreement between all the measures as for the number of clusters. Prior work that used automated priming approach showed usefulness of combined indices in discovery of interesting clusters in patient discharge summaries [1,13,14].

Automated priming can be explained by a spreading activation theory [12]. It gives mathematical simplicity therefore it was used here for modeling the priming effect. At the initial step $t = 0$ of the spreading activation only a subset columns in the M matrix have non-zero elements. As activation spreads from initial concepts to semantically related concepts the number of non-zero columns M increases. The initial concept space representation is symbolized here by ${}_0M$. Let's assume that R is a symmetric matrix with all possible semantic relations between MeSH terms, derived from the UMLS. Then $R_i^j = R_j^i = 1$ means that there is a semantic relation (e.g. *is_a, is_associated_with, may_be_treated_by*, etc.) between concept i and a concept j (e.g. *cerebral structure is_finding_site_of alzheimer's disease*). The simplest mathematical model of spreading activation scheme can be written as:

$${}_{t+1}M = \mathrm{f}({}_tM + \mathrm{f}({}_tMR)). \tag{2}$$

This process enhances the previous feature space ${}_tM$ with new, semantically related concepts, defined by the R matrix. Simple spreading activation algorithms have been already investigated by [24,25,26,27,28]. These approaches did not exploit the full potential of semantic networks (only parent/child relationships were used).

On the other hand if the full potential of semantic relations from UMLS are used for spreading activation it may result in associations similar to that in a schizoid brain (almost everything is associated with everything) [29]. One solution to this problem is to inhibit certain relations by removing those concepts that do not help in assigning documents into right clusters. They may be roughly identified using feature ranking techniques. Ranking is done using Legendre distance (Eq. 1) between columns in the ${}_{t+1}M$ matrix and the cluster labels. The main difference of this technique from a standard feature filtering algorithm is that the ranking is done for each of the cluster separately. So if there are three clusters identified then there are three sets of ranking. The best concepts are taken from each ranking and are put in a set that will excite only those concepts that passed ranking test. This set is called here ${}_{t+1}\mathcal{E}$ and can be roughly defined by:

$$ {}_{t+1}\mathcal{E} = \bigcup_i \text{ the best concepts according to eq. 1 that represent } i\text{th cluster.} \tag{3}$$

The original adjacency matrix R was inhibited to a ${}_tR'$ matrix according to a rule:

$$ {}_tR' = \begin{cases} 0 & \text{if } j \notin {}_{t+1}\mathcal{E} \\ R_{ij} & \text{otherwise} \end{cases} . \tag{4}$$

Enhanced with additional semantic knowledge matrix ${}_{t+1}M$ was calculated using inhibited ${}_tR'$ matrix and the simple neuron threshold output function:

$$ {}_{t+1}M = \mathrm{f}({}_tM + \mathrm{f}({}_tM_tR')). \tag{5}$$

Semantic graphs are computed using high quality PubMed query results clusters. These graphs are called *graphs of consistent concepts* (GCC) [30]. The idea of GCC that represents a PubMed query results is to show an optimal number of concepts that represent each query cluster. There should be maximum number of connection between concepts that belong to the same cluster and minimum number of connection between concepts that represent different clusters. Increasing the number of concepts that represent each cluster increases also chance that the lower rank concept will have semantic connections with concepts that represent other clusters.

First step is to rank concepts for each cluster separately using Legendre distance from eq. 1. Next a function is defined:

$$ \mathrm{Concepts}(i,n) = \text{ a set of } n \text{ best concepts using eq. 1 that represent } i\text{th cluster.} \tag{6}$$

In order to make the GCC optimization process simple n was varied the same way for each cluster. This gives a set of of concepts that will be used to create adjacency matrix:

$$ \mathcal{E}^n = \bigcup_i \mathrm{Concepts}(i,n) \tag{7}$$

After $iter$ number of iterations of clustering and spreading activation the R matrix can be modified according to following rule to create an adjacency matrix for the semantic graphs:

$$ {}^{iter}_{n}R = \begin{cases} R_{ij} & \text{if } i \in \mathcal{E}^n \text{ and } j \in \mathcal{E}^n \\ 0 & \text{otherwise} \end{cases} . \tag{8} $$

At the beginning n, the best n representants for each cluster, was unknown. It was computed maximizing following function:

$$ \begin{aligned} \mathrm{gcc}({}^{iter}_{n}R) = & \sum_{k} \left(\sum_{\{i|\mathrm{Concepts}(i,n)\}} \sum_{\{j|\mathrm{Concepts}(j,n)\}} {}^{iter}_{n}R_i^j \right) /2o \\ & - \sum_{\substack{l,m \\ l \neq m}} \left(\sum_{\{i|\mathrm{Concepts}(i,n)\}} \sum_{\{j|\mathrm{Concepts}(j,n)\}} {}^{iter}_{n}R_i^j \right) /2o + p/o \end{aligned} \tag{9} $$

where p is the number of clusters, o is the number of active concepts ($o = |\mathcal{E}^n|$). The first term of this equation sums relations between concepts that represent the same cluster, second term sums relations between concepts representing different clusters, while the last term adds the number of clusters. All terms are divided by the total number of concepts o to assure that $\mathrm{gcc}({}_{n}R) = 1$ when all concepts representing the same clusters are connected by a minimum spanning trees and there are no connections between concepts that represent different clusters.

A summary of the algorithm for creating semantic graphs that can be used as input to the VL prototype can be written as a pseudo-code (algorithm 1). The algorithm takes as an *input* data matrix, semantic relations, a number of iterations and *outputs* optimal GCC.

Algorithm 1. GCC algorithm

```
function GCC($R$, ${}_0M$, $iterations$)
    for $t$ in 0 to $iter$ do
        compute distance matrix based on ${}_tM$ matrix
        compute tree based on distance matrix and Ward's clustering algorithm
        compute number of clusters based on combined and normalized cluster quality index
        compute the best ranked features separately for each cluster that will be activated
        compute ${}_tR'$ by inhibiting original ${}_tR$ matrix using the best ranked features
        compute ${}_{t+1}M$ based on inhibited ${}_tR'$, ${}_tM$ and neuron output function
    end for
    compute ranking of features for each cluster separately
    compute $n$ best ranked features that maximize graph consistency index
    compute ${}^{iter}_{n}R$ adjacency matrix with optimal number $n$ best ranked features
    return graph of consistent concepts
end function
```

4 Results

Concepts that occurred in less than three documents, and documents that had less then three concepts were removed. This created a matrix with 2575 documents and 416 concepts. Two constrains were applied to the system: concepts that receive activation from at least two other concepts are activated (threshold of the f(·) function is set to 2), and semantic relations are used if they are mentioned in at least two UMLS sources (e.g. MeSH and SNOMED CT).

Six steps of spreading activation were applied to the initial data matrix. The feature space increased from 416 to only 423 concepts. Every step of spreading activation increased quality of clusters for all measures: Davies-Bouldin's index was reduced from 2.2844 to 1.3849, pseudo F-Statistic was increased from 112. 96 to 940.15, Hubert-Levine index was reduced 0.4574 to 0.4414, and Silhouette cluster quality was increased from 0.1515 to 0.3867.

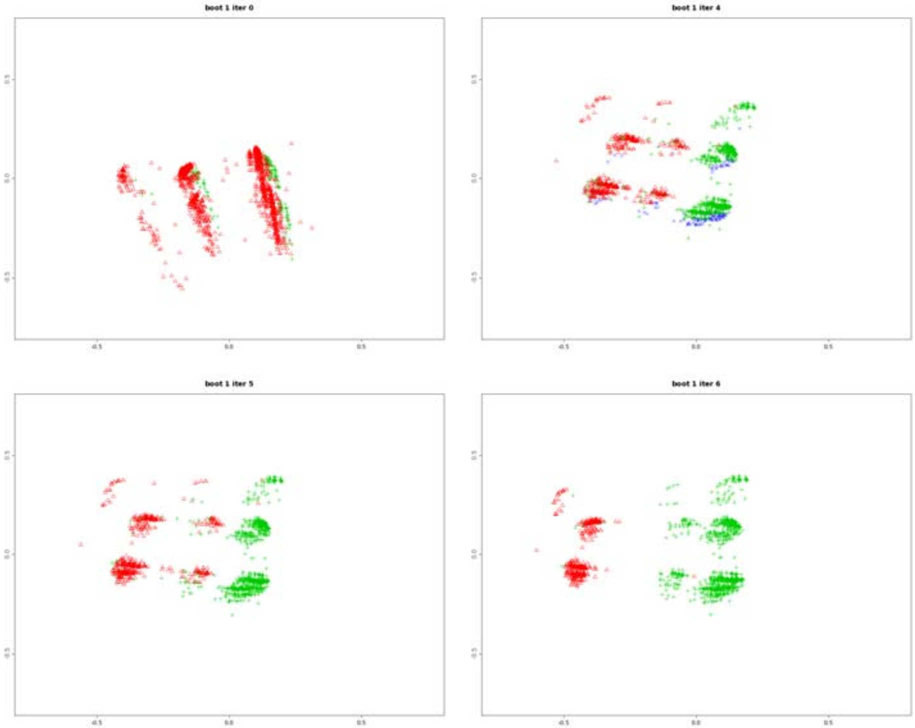


Fig. 2. Multidimensional scaling for *"Alzheimer disease"[MeSH Terms] AND "apolipoproteins e"[MeSH Terms] AND "humans"[MeSH Terms]* PubMed query at the initial step ($t = 0$), after fourth ($t = 4$), fifth ($t = 5$) and sixth step ($t = 6$) of spreading activation. Red triangle sign (first cluster), green plus sign (second cluster) and blue x sign (third cluster) show to which cluster a document concept belongs to. Each step of feature space enhancement creates clearer clusters.

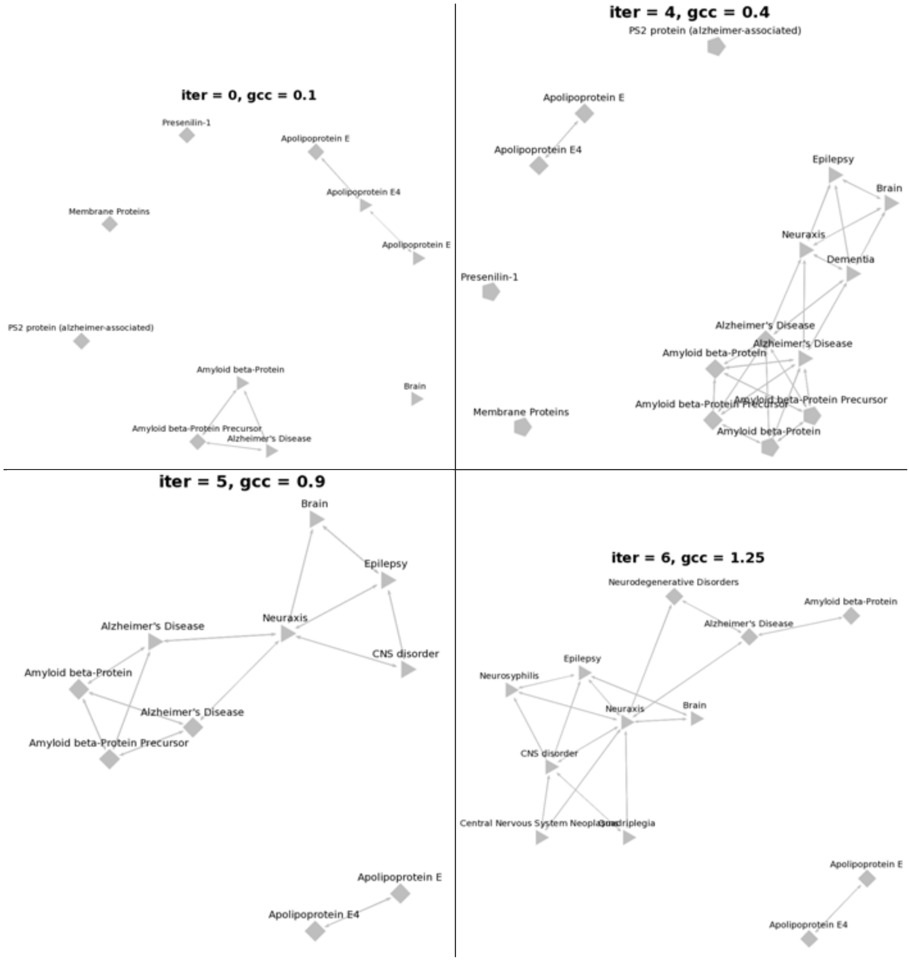


Fig. 3. Semantic graphs for *"Alzheimer disease"[MeSH Terms] AND "apolipoproteins e"[MeSH Terms] AND "humans"[MeSH Terms]* PubMed query at the initial step ($t = 0$), after fourth ($t = 4$), fifth ($t = 5$) and sixth step ($t = 6$) of spreading activation. Triangle (first cluster), rectangle (second cluster) and pentagon (third cluster) nodes show to which cluster a concept belongs to (see eq. 6). Each image shows the optimal value of *gcc* function (see eq. 9).

The most important finding is that the initial graph that represents query clusters has low maximum consistency measure $\mathrm{gcc}({}^{0}_{n}R) = 0.1$. Adding new information that simulates associative processes in the expert's brains using spreading activation networks increases not only the cluster quality but also consistency of semantic graph that represents same query results clusters $\mathrm{gcc}({}^{6}_{n}R = 1.25)$. Fig. 3 shows the increase of the consistency measure calculated using eq. 9 while fig. 2 shows changes in the 2D projections of the high dimensional concept space using classic multidimensional scaling.

5 Conclusion

In computational intelligence community WebSOM [31] has been the most well-known attempt to visualize information using clustering techniques. However, in many respects 2-dimensional visual representation of hierarchical visualization of clusters offered by SOM is not sufficient: relations between documents are much more complicated and may only be shown in a non-planar graphs. Instead presentation of query results that match human cognitive abilities in a best way may be done using neurocognitive inspirations.

The algorithm presented here tries to re-create pathways of neural activation in the expert's brain, enhancing the initial concepts found in the text using relations between the concepts. This may be done in the medical domain because specific relations between MeSH terms present in a citation may be extracted from the huge UMLS resources [10]. Background knowledge is added using adjacency matrix describing semantic relations. The whole algorithm is presented in the matrix form, making it suitable for efficient large-scale retrieval systems. The goal here is to automatically create graphs of consistent concepts using documents that result from specific queries, and present these concepts using Visual Language iconic system [7]. This paper has demonstrated the usefulness of neurocognitive inspirations approximating brain processes with a combination of clustering, feature selection and neural spreading activation techniques. While there is an ample room for improvement using better clustering techniques and feature selection methods a significant increase of the quality of the initial GCC graph has been obtained, giving much better representation of the information in a visual form.

References

1. Duch, W., Matykiewicz, P., Pestian, J.: Neurolinguistic approach to natural language processing with applications to medical text analysis. Neural Networks (in press, 2008), doi:10.1016/j.neunet.2008.05.008
2. Buzan, T.: The Mind Map Book. Penguin Books (2000)
3. Farrand, P., Hussain, F., Hennessy, E.: The efficacy of the mind map study technique. Medical Education 36(5), 426–431 (2002)
4. Kraft, R., Zien, J.: Mining anchor text for query refinement. In: Proceedings of the 13th international conference on World Wide Web, pp. 666–674. ACM, New York (2004)
5. Gordon, M.D., Lindsay, R.K.: Toward discovery support systems: a replication, re-examination, and extension of swanson's work on literature-based discovery of a connection between raynaud's and fish oil. J. Am. Soc. Inf. Sci. 47(2), 116–128 (1996)
6. Zender, P.M., Crutcher, K.A.: Visualizing alzheimer's disease research: a classroom collaboration of design and science. In: SIGGRAPH 2004: ACM SIGGRAPH 2004 Educators program, p. 24. ACM, New York (2004)
7. Zender, M., Crutcher, K.A.: Visual language for the expression of scientific concepts. Visible Language 41, 23–49 (2007)
8. Duch, W., Szymaski, J.: Semantic web: Asking the right questions. In: Gen, M., Zhao, X., Gao, J. (eds.) Series of Information and Management Sciences, California Polytechnic State University, CA, USA, pp. 456–463 (2008)
9. Pedrycz, W.: Knowledge-Based Clustering: From Data to Information Granules. Wiley Interscience, Hoboken (2005)

10. U.S. National Library of Medicine, National Institutes of Health: Unified medical language system (January 2007), http://www.nlm.nih.gov/research/umls/
11. Zender, M.: Advancing icon design for global non verbal communication: Or what does the word bow mean? Visible Language 40, 177–206 (2006)
12. McNamara, T.P.: Semantic Priming: Perspectives From Memory and Word Recognition. Psychology Press, Taylor & Francis Group (2005)
13. Duch, W., Matykiewicz, P., Pestian, J.: Towards Understanding of Natural Language: Neurocognitive Inspirations. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D.P. (eds.) ICANN 2007. LNCS, vol. 4669, pp. 953–962. Springer, Heidelberg (2007)
14. Duch, W., Matykiewicz, P., Pestian, J.: Neurolinguistic approach to vector representation of medical concepts. In: Press, I. (ed.) Proc. of the 20th Int. Joint Conference on Neural Networks (IJCNN), August 2007, p. 1808 (2007)
15. Jurasinski, G.: simba: A Collection of functions for similarity calculation of binary data, R package version 0.2-5 (2007)
16. Legendre, P., Legendre, L.: Numerical Ecology. Elsevier, Amsterdam (1998)
17. Anderberg, M.R.: Cluster Analysis for Applications. Academic Press, New York (1973)
18. Milligan, G.W.: An examination of the effect of six types of error perturbation on fifteen clustering algorithms. Psychometrika 45, 325–342 (1980)
19. Walesiak, M., Dudek, A.: clusterSim: Searching for optimal clustering procedure for a data set, R package version 0.36-1 (2008)
20. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence 1, 224–227 (1979)
21. Calinski, R.B., Harabasz, J.: A dendrite method for cluster analysis. Communications in Statistics 3, 1–27 (1974)
22. Milligan, G.W., Cooper, M.C.: An examination of procedures of determining the number of cluster in a data set. Psychometrika 50, 159–179 (1985)
23. Kaufman, L., Rousseeuw, P.J.: Finding groups in data: an introduction to cluster analysis. Wiley, New York (1990)
24. Hotho, A., Staab, S., Stumme, G.: Wordnet improves text document clustering. In: Proc. of the Semantic Web Workshop at SIGIR-2003, 26th Annual International ACM SIGIR Conference (2003)
25. Sedding, J., Kazakov, D.: Wordnet-based text document clustering. In: Pallotta, V., Todirascu, A. (eds.) COLING 2004 3rd Workshop on Robust Methods in Analysis of Natural Language Data, Geneva, Switzerland, August 2004, pp. 104–113 (2004)
26. Struble, C.A., Dharmanolla, C.: Clustering mesh representations of biomedical literature. In: Lynette, H., James, P. (eds.) HLT-NAACL 2004 Workshop: BioLINK 2004, Linking Biological Literature, Ontologies and Databases, May 2004, pp. 41–48. Association for Computational Linguistics, Boston (2004)
27. Yoo, I., Hu, X., Song, I.-Y.: A coherent biomedical literature clustering and summarization approach through ontology-enriched graphical representations. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2006. LNCS, vol. 4081, pp. 374–383. Springer, Heidelberg (2006)
28. Jing, L., Zhou, L., Ng, M.K., Huang, J.Z.: Ontology-based distance measure for text clustering. In: Proceedings of the IV Workshop on Text Mining; VI SIAM International Conference on Data Mining (April 2006)
29. Kreher, D., Holocomb, P., Goff, D., Kuperberg, G.: Neural evidence for faster and further automatic spreading activation in schizophrenic thought disorder. Schizophrenia bulletin 34, 473–482 (2008)
30. Matykiewicz, P., Duch, W., Pestian, J.: Nonambiguous concept mapping in medical domain. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 941–950. Springer, Heidelberg (2006)
31. Kohonen, T., Kaski, S., Lagus, K., Salojrvi, J., Paatero, V., Saarela, A.: Organization of a massive document collection. IEEE Transactions on Neural Networks 11(3), 574–585 (2000)

Search-In-Synchrony: Personalizing Web Search with Cognitive User Profile Model

Chandra Shekhar Dhir[1,3,⋆] and Soo Young Lee[1,2,3]

[1] Department of Bio and Brain Engineering
[2] Department of Electrical Engineering and Computer Science
[3] Brain Science Research Center
KAIST, Daejeon 305-701, Korea
Phone: +82-42-869-5431; Fax: +82-42-869-8490
shekhardhir@neuron.kaist.ac.kr

Abstract. This study concentrates on adapting the user profile model (UPM) based on individual's continuous interaction with preferred search engines. UPM re-ranks the retrieved information from World Wide Web (WWW) to provide effective personalization for a given search query. The temporal adaptation of UPM is considered as a one-to-one socio-interaction between the dynamics of WWW and cognitive information seeking behavior of the user. The dynamics of WWW and consensus relevant ranking of information is a collaborative effect of inter-connected users, which makes it difficult to analyze in-parts. The proposed system is named as Search-in-Synchrony and a preliminary study is done on user group with background in computational neuroscience. Human-agent interaction (HAI) can implicitly model these dynamics. Hence, a primary attempt to converge the two fields is highlighted - HAI and statistically learned UPM to incorporate cognitive abilities to search agents.

1 Introduction

Effective and efficient information retrieval (IR) from the vast amount of data present in the WWW is of primary importance to the day-to-day internet users. Personalization of web searches shifts the domain from consensus relevancy (one model fits the entire populations) to personal relevancy [2]. This shift in IR from WWW can help in filtering out websites which do not interest the user, thereby, reducing the search time and user's effort (usage of computer peripherals). A number of researches have been actively directed towards personalization of web searches based on UPM [2].

Personalization of web searches can be accomplished by a user-adaptive search agent which works between the user and popular search engine. A personalized search agent computationally incorporates user's desire by exploiting *apriori* knowledge on their IR behavior and cognitive search strategies. A number of

⋆ Corresponding author.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 80–87, 2009.

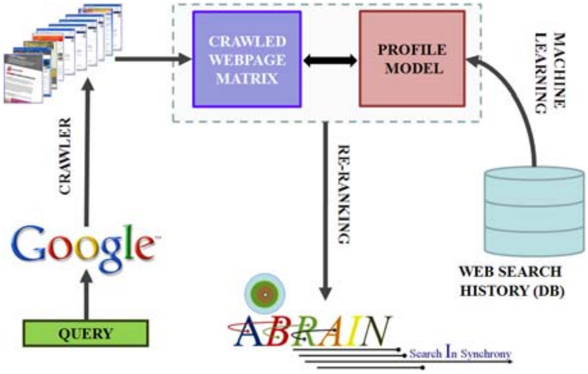


Fig. 1. Architecture of proposed Search-in-Synchrony

machine learning algorithms can model user's IR behavior while using popular internet search engines [3,4]. Neural network approaches can also be used to categorize webpages in document clusters using content similarity [5]. Although, user profile model can enhance effective IR from the WWW, design of consistent UPM is a challenging task considering the dynamics of human interest. In addition, UPM must be flexible to temporal adapt with the dynamics nature of WWW and other contextual factors. In literature, such intelligent internet agents are referred as reconnaissance agents [6]. Reconnaissance agents perform contextual computing which computationally models the user behavior and also adapts to its multi-user environment; thus, depicting social cognition [7]. Web-based search behavior follows implicit characteristics of its user. Ford et al. argued the existence of symbiotic relationship among inter-personal web searching strategies and human individual difference. The individual differences are primarily due to their study approaches, cognitive and demographic features, internet perception and information seeking behavior on WWW [8]. Information seeking behavior and cognitive search strategies can also be utilized in computational designs of personalized agents to highlight information seeking behavior of users [9].

The proposed architecture of Search-in-Synchrony is shown in figure 1 and discussed in section 2. Section 3 presents the re-ranking criteria for effective personalization of web search experience. In section 4, preliminary results compare the performance of proposed Search-in-Synchrony with results of Google search [16]. The paper is concluded in section 5.

2 Search-In-Synchrony - Architecture of Proposed Personalized Search Agent

Once the user gives a query using the web-interface [15], top L Page-ranked [10] webpages from Google searches are crawled. The content of the web pages appearing in the order of their Page-rank is processed by UPM to re-rank search results in coherence with user's satisfaction. In a way, we are trying to train UPM to mimic user's behavior while retrieving data from WWW - *a personalized search agent that works like you.*

Web search history database. Data collection is important to initialize the personalized search agent, UPM. A toolbar is designed as an interface between the user and web search engines (in our study Google search is used). Once a search query is submitted, users are directed to the original Page-ranked webpages retrieved by Google search. If the particular webpage is of interest, users can store it's content by clicking on the *interest* button on the toolbar. Clicking *interest* or $not - interest$ button on toolbar initiates parsing of HTML contents of the webpage being surfed. The parsed text data is stored in the web search database along with search query, web URL, and user name. Binary categorization of surfed webpages in database can be utilized to filter irrelevant information from WWW and present more personalized search results. A case study is done with the students of Computational NeuroSystems Lab (CNSL) at KAIST [1]. The web search history database is constructed by using the webpages relevant to general research interests in Computational Neuroscience on hyperlink ontology of Open Directory Project [11], Wikipedia [12] and Wikibooks [13]. In addition, user group with background in computational neuroscience update the database with websites relevant to their research interests.

User profile model. Before constructing the UPM, text data corresponding to webpages in the database is preprocessed:

- Removal of non-alphabetical characters like ϵ, 1, etc.
- All alphabetical characters are made lower case.

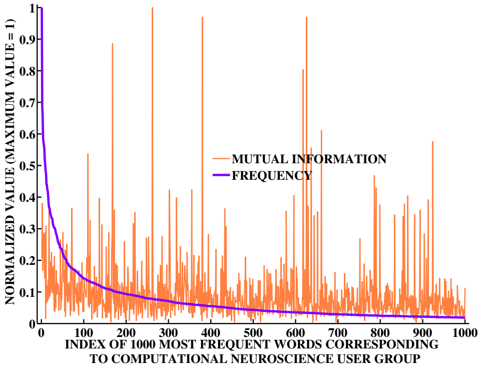


Fig. 2. Mutual information measure and frequency of the 1000 most frequent words extracted from webpage relevant to user's interest in web search database. In this case study, user's interest is in computational neuroscience.

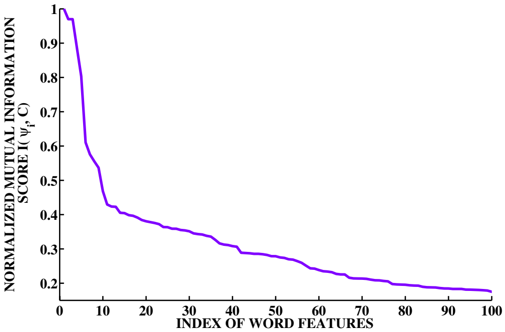


Fig. 3. Decay of mutual information of 100 most significant features (words) extracted from computational neuroscience web search database

- Stop-word removal: all stop words (ex. *a*, *the*, etc.) are filtered from the text database to increase search performance.
- Removal of words with length ≤ 2.

After pre-processing, UPM is designed by selecting K word features with maximal mutual information from F most frequent words extracted from the webpages of user's interest in the search database. Mutual information criteria is a machine learning approach which is defined as [14],

$$I(\phi_i; C) = \sum_{c \in c_1}^{c_2} \sum_{\phi_i} p(\phi_i, c) \frac{p(\phi_i, c)}{p(\phi_r)p(c)}, \tag{1}$$

where $p(\cdot)$ is the probability distribution of the input random variable and ϕ_i is the i-th most frequent word feature. $|\cdot|$ is defined as the cardinality of the input set. c_1 is the category of webpages that are of interest to the user, while c_2 includes the remaining websites which are of minimal interest to the user. Using the value of $I(\phi_i, C)$, we can select the i-th most significant word feature, ψ_i, as

$$\begin{aligned} r &= \arg\max_i I(\phi_i; C), \\ \psi_i &= \phi_r, \quad 1 \leq i, r \leq |c_1|. \end{aligned} \tag{2}$$

Figure 2 shows the mutual information measure and frequency of the 1000 most frequent words.

UPM is a word-weblink 2-D matrix, $\mathbf{W}_{K \times |c_1|} = [\mathbf{w}_1 \cdots \mathbf{w}_j \cdots \mathbf{w}_{|c_1|}]$, where $\mathbf{w}_j = [w_{1j} \cdots w_{ij} \cdots w_{Kj}]^T$. w_{ij} is the frequency of occurrence of i-th most significant word features, ψ_i in j-th webpage and $[\cdot]^T$ is the transpose operator in input matrix. In this case study on users with sound background in computational neuroscience, $F = 1000$, $D = 862$, $K = 100$, $|c_1| = 315$ and $|c_2| = 547$. Figure 3 shows the decay of $I(\phi_i, C$ for the most significant 100 features (words). UPM ($\mathbf{W}_{100 \times 315}$) is shown in figure 4.

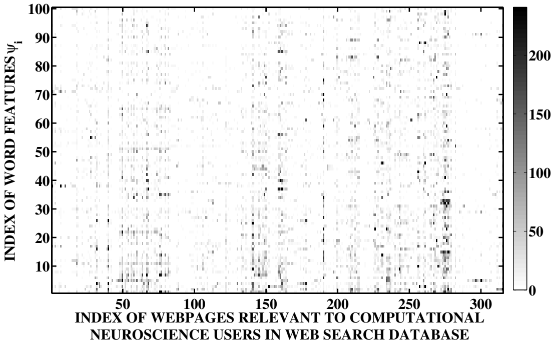


Fig. 4. User Profile Model: 2-D word-weblink matrix constructed from web search database. Elements of the matrix show the frequency of occurrence of 100 words selected from 1000 most frequent words based on their mutual information with webpages of interest and not-interest.

3 Re-ranking of Page-Ranked Results Using User Profile Model

Top L-Page-ranked webpages crawled for a given query are re-ranked using the UPM. The content of crawled webpages is pre-processed as discussed in section 2. To improve the computational speed stop words are not removed from the text content of the crawled webpages.

The crawled webpage matrix for given search query is a 2-D word-frequency matrix, $\mathbf{V} = [\mathbf{v}_1 \cdots \mathbf{v}_l \cdots \mathbf{v}_{|c_1|}]$ where $\mathbf{v}_l = [v_{1l} \cdots v_{il} \cdots v_{Kl}]^T$. v_{il} is the frequency of occurrence of ψ_i in the l-th Page-ranked webpage. Figure 5 shows the crawled webpage matrix for search query SVM (referred as Support Vector Machines in computational neuroscience area).

Both UPM and crawled webpage matrix are utilized to re-rank the l-th PageRanked webpage. The re-ranking criteria is defined as,

$$rank(l) \propto \sum_{i=1}^{|c_1|} M(\mathbf{w}_i, \mathbf{v}_l), \tag{3}$$

where $M(\mathbf{w}_i, \mathbf{v}_l)$ is the distance metric. In this case study, cosine similarity between $\mathbf{w}_i$ and $\mathbf{v}_l$ is used as the distance metric.

Figure 6 shows the re-ranking of the Page-ranked webpages by Search-in-Synchrony for the search query SVM. Webpages which are of interest to the user shows better ranking with personalized search agent as seen from the cluster of red squares in figure 6. Latency and user efforts are greatly reduced if webpages of interest can be among the top 20 searched webpages.

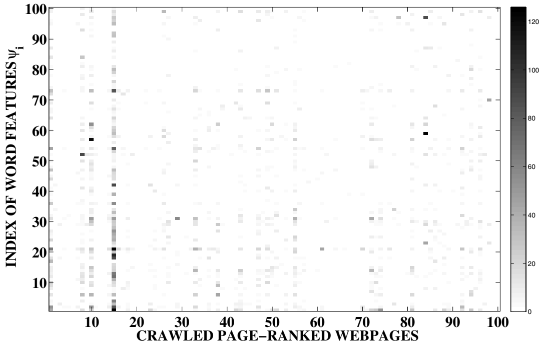


Fig. 5. Crawled webpage matrix for search query SVM. x-axis shows the index of top 100 Page-ranked webpages for search query SVM (in decreasing order of Page-rank).

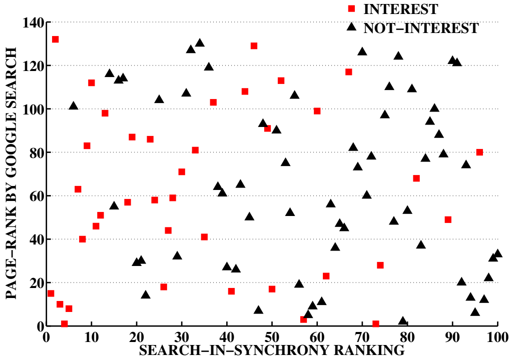
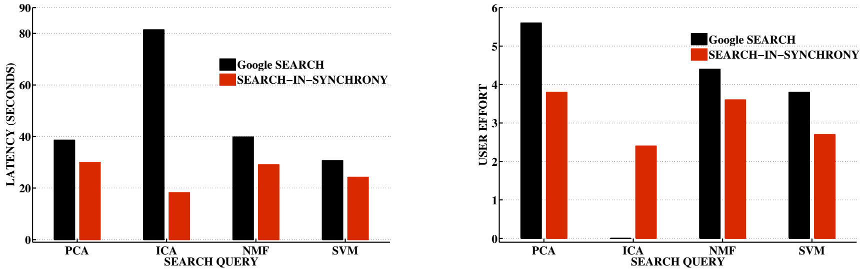


Fig. 6. Re-ranked results by Search-in-Synchrony for search query SVM. Red square refer to webpages of interest to computational neuroscience researcher and black triangle refer to webpages which are of minimal interest.

4 Performance Evaluation of Search-in-Synchrony

The performance of personalized search agent is evaluated based on two parameters – a) latency and b) effort (additional usage of computer peripherals like mouse and keyboard) in finding information from WWW which satisfies user's interest. For the case study, 10 computational neuroscience researchers at CNSL were asked to compare their web search experience for Search-in-Synchrony and Google search. Supervised web search for the following queries were made by each subject: a) PCA – Principal Component Analysis, b) ICA – Independent Component Analysis, c) NMF – Non-negative Matrix factorization, and d) SVM – Support Vector Machines.

For the given search query, subjects were asked to find sufficient information using Google search until they were satisfied. In similar manner, they were also asked to search relevant webpages using Search-in-Synchrony for the same search query. Figure 7 shows that the average latency and effort of the subjects in



(a) Latency

(b) User effort

Fig. 7. Average a) time taken, b) number of mouse clicks by 10 subjects to find relevant webpages corresponding to search query. In some cases, subjects gave up searching for ICA using Google search engine, therefore, the average of additional mouse clicks for the case is not given.

searching relevant information from WWW for the given query is reduced when Search-in-Synchrony is used instead of Google search. Especially, for the search query ICA average latency is much less, in addition, most users gave up retrieving information using Google search as the relevant results did not appear on first 5 pages (going from 1$^{\mathrm{st}}$ search page to other is time consuming and requires additional effort from the user).

5 Conclusions

The proposed model, Search-in-Synchrony, personalizes information retrieval from World Wide Web. It powers web search experience by reducing latency and effort to retrieve information which maximally coheres with user's interest. UPM which is at the heart of Search-in-Synchrony is a reconnaissance agent working on HAI philosophy. It can effectively model user's cognitive web search behavior by implicitly extracting features relevant to their interest. Mutual information between user's IR behavior and their interests is exploited for supervised selection of relevant information features for personalized WWW search.

Initial results on small population with similar interests shows motivating improvements in latency and efforts during IR from WWW. Application of the proposed model on large population set is currently being explored for improved generalization and commercialization of personalized web search.

References

1. http://cnsl.kaist.ac.kr/
2. Pitkow, J., et al.: Personalized search. Communication of the Assoc. of Computing Machinery (2002)

3. Lerner, B., Lawrence, N.: A comparison of state-of-theart classification techniques with application to cytogenetics. Neural Computing and Applications, 39–47 (2001)
4. Williams, A., Ren, Z.: Agents teaching agents to share meaning. In: Proc. Fifth Intl. Conf. on Autonomous Agents, pp. 465–472 (2001)
5. Liu, Z., Zhang, Y.: A competitive neural network approach to Web-page categorization. Intl. Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 9, 731–741 (2001)
6. Lieberman, H., et al.: Exploring the web with reconnaissance agents. Communications of the ACM 44, 69–75 (2001)
7. Richards, M.: Internet learning agents: a study of user performance with selected search engines. In: IEEE Proc. SoutheastCon, pp. 437–444 (2005)
8. Ford, N., et al.: Web search strategies and human individual differences: A combined analysis. Journal of the American Society for Information Science and Technology 56(7), 757–764 (2005)
9. Thatcher, A.: Information-seeking behavior and cognitive search strategies in different search tasks on the WWW. Intl. Jounral of Industrial Ergonomics 36(12), 1055–1068 (2006)
10. Page, L., et al.: The PageRank Citation Ranking: Bringing Order to the Web. Stanford Digital Library Technologies Project (1998)
11. http://www.dmoz.org/
12. http://www.wikipedia.org/
13. http://en.wikibooks.org/wiki/Wikibooks_portal
14. Dhir, C.S.: Efficient Feature Selection Based on Information Gain Criterion for Face Recognition. In: IEEE Intl. Conf. on Information Acquisition (2007)
15. http://crldec4.kaist.ac.kr/psearch/search.py
16. http://www.google.com/

Neurocognitive Approach to Creativity in the Domain of Word-Invention

Maciej Pilichowski[1] and Włodzisław Duch[2]

[1] Faculty of Mathematics and Computer Science
[2] Department of Informatics, Nicolaus Copernicus University, Toruń, Poland
macias@mat.umk.pl, Google: W. Duch

Abstract. One of the simplest creative act is the invention of a new word that captures some characteristics of objects or processes, for example industrial or software products, activity of companies, or the main topic of web pages. Neurocognitive mechanisms responsible for this process are partially understood and in a simplified form may be implemented in a computational model. An algorithm based on such inspirations is described and tested, generating many interesting novel names associated with a set of keywords.

1 Introduction

Creativity understood as "the capacity to create a solution that is both novel and appropriate" [1], manifests itself not only in creation of novel theories or inventions, but permeates our everyday life, including comprehension and production of text and speech. The best chance to capture creativity in computational algorithms is to follow neurocognitive inspirations [2,3].

Despite the limitation of the current knowledge of the neural processes that give rise to the higher cognitive processes in the brain it is possible to propose a testable, neurocognitive model of creative processes. A review of the psychological perspectives and neuroscientific experiments on creativity has already been presented in [2,3]. Three elements are essential for computational model of creativity: first, a neural space for encoding probabilities of strings of letters and phonemes in a given language, leading to automatic associations of keywords with semantically and phonetically related words through spreading of neural activation; second, imagination based on combinations of phonemes or letters, with probability that depends on the priming effects [4]; and third, the filtering of interesting results based on density of phonological and semantic associations. Creative brains are well trained, have greater imagination, combine faster basic primitives, and pick up interesting combinations of these primitives through emotional and associative filtering. For novel words filters based on phonological and semantic plausibility may be proposed. Phonological filters may be realized using autocorrelation matrix memories or probabilistic models. Semantic filters should evaluate how many words have similar meaning to the new word, including similarity to morphemes that the words may be decomposed to.

The Mambo algorithm described here is one of many possible models based on these principles. It is tested on the invention of novel, interesting words that are suitable as names for products, companies, web sites or names of various entities.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 88–96, 2009.

2 Algorithmic Considerations

Initially the Mambo system does not have any *a priori* linguistic knowledge. Google Web 1T 5-gram [5] dictionary is used as the training source. This dictionary is based only on word frequency, therefore it is necessary to perform at least minimal spell-checking before using this data for training. The LRAGR [6] and SCOWL [7] dictionaries have been used to correct the misspelled words. An unexpected and quite surprising problem is the relative shallowness of such large dictionary provided by Google – over 40% of words that exist in standard English dictionaries are not yet present in the Google dictionary, probably due to the large number of rare and outdated words in English [8].

Dictionary presented to the system is used to learn morphological regularities using probabilistic model. This model is somewhat easier to control than the autocorrelation matrix memory that may also be used for such learning. Word frequencies are used in the learning process. However, using raw word frequencies would lead to very frequent presentation of some common words (mostly articles and conjunctions). For example "the" occurred 23 billion times, while "of", the next most common word, occurred 13 billion times only. On the other end of the list such word as "viscoses" has frequency around 200. To avoid these huge differences logarithmic scale for frequencies is used.

Filtered data (words and their frequencies) are the source of knowledge for the system – depending on selected symbol set, data can be presented as letters ("the" → "t", "h", "e"), phonemes ("the" → "ð", "ə"), in a semi-letter form ("the" → "th", "e"), only for English, or in a mixed form (a word is processed in several forms at the same time). In our implementation mixed mode with letters and phonemes causes huge CPU load because conversion to a phoneme representation is done via a slow external program. The pure phoneme form, written using the IPA (International Phonetic Alphabet) codes, is hard to read for most people. The semi-letter form is easily readable and as fast to process as the letter one, allowing to keep the atomicity of most characteristic phonemes ("sh", "ch", "th", "ph", "wh", "rh").

Semantics. The system learns the structure of the words but not their meanings. Thus it does not have such flexible associations as people. For example, "light" may mean "small weight", but it may also be associated with "daylight". The program is unaware of intended or unintended collocations of symbol strings. Advertising some product it is helpful to associate "possibilities" with "great" (positive association), rather than "problems" (negative association). Currently the Mambo system does not include word valence, but such an extension is not difficult, as affective word lists are readily available. Without it in real applications better results are obtained if a distinct sets of priming words with only negative associations ("bound", "less"), or words with only positive associations ("more", "energy"), are provided, but not both sets.

Associations. In a typical situation a description of the product will be given, read, and names for this product will be proposed by a human expert. Reading such description leads to priming of the network, making it more susceptible to activation for the same and related words. Priming is one of the effects of implicit memory which increases the probability of recalling the previously encountered item, not necessarily in the same form (as defined by Kalat [9]). Priming is achieved by repetition of the presented item or by presenting items associated with the target [4].

It is not easy to implement realistic priming in an automatic way. WordNet database [10] that collects words with similar meanings into "synsets" may help, but despite extensive efforts the content of WordNet is still far from perfect, and even for simple queries questionable associations arise. Therefore WordNet is used only to help the user to prepare the priming set.

Word determinant (prefix or suffix) is defined as the shortest prefix (suffix) of word which is not a prefix (suffix) of any other word in the dictionary, with the exception of derivatives of this word. For example, the prefix determinant of "education" is "educat" ("educated", "educational", and other forms are derivates of "educate"). Possible backward associations may become a problem: the system can create a word which is relevant to the priming set, but this word may be related in a much stronger way to words outside of the priming set. As an example consider the priming item "owl", and the system's association with more frequent "bowl" that may dominate [11] inhibit correct associations to "owl". There is no good mechanism that could control such hijacking, the system can only enforce existence of word determinants from the priming set. This enforcement has diminished the problem but did not eliminate it completely. Restricting created words to those that contain at least one word determinant will assure that results will be associated with the desired (priming) word set.

Originality. Words created by the system should be original, therefore copies of previously learnt words should be avoided. At the filtering level all shortened forms of already known words are rejected ("borrow" → "borr") and words with superficial changes ("borrow" → "borrom"). At first this looks like an obvious decision, but for real world application it may be too strict – just consider "sorrow", "barrow", or "burrow". Originality plays the most important role in applications of Mambo system. Brand names or company names should distinguish themselves from those already existing (although similarity has some advantages, for example several "Imtel" or "Indel" companies exist). The same rules are also used in order to avoid numerous variations on a single created word.

Imitations. Compound words, such as the "bodyguard", "brainstorm" or "airmail", are highly ranked by the Mambo system. While testing the system on concepts related to aviation quite compelling word "jetmail" has been created, but only because the system was primed with the word "jet" and there was word "• • •mail" in the dictionary. The first part gave it a high association rank, the latter contributed to good linguistic quality of the result. Thus creation of compound words in the Mambo system may lead to a form of plagiarism – the system is creative only if humans were creative before. Although this may be corrected in the final stage (searching the Internet to determine if the word is new), for the purpose of testing all compound words have been removed from the dictionary to guarantee that words created by the Mambo system result from its internal algorithm, and not from a simple word takeover.

3 Word Rank Function

Words are ranked by their linguistic (phonological) and semantic quality, depending on associations that they may invoke. The word rank function $Q'(w)$ is defined as:

$$Q'(w) = \prod_{i=0}^{|ng(T(w))|-1} q(ng(T(w))[i]) \tag{1}$$

where function q is a dictionary function, for a given ngram returning numerical value based on previously processed dictionary data (see below), $T(w)$ is a composition of word w transformations, ng is a function partitioning symbols in w into overlapping ngrams of length $N_{ng}+1$, shifted by S_{ng}. Function $ng(w)$ returns a sequence of strings (ngrams):

$$\begin{aligned} ng(w) = \langle & w[0 : N_{ng}], w[S_{ng} : S_{ng} + N_{ng}], w[2S_{ng} : 2S_{ng} + N_{ng}], ..., \\ & w[nS_{ng} : nS_{ng} + N_{ng}]\rangle \end{aligned} \tag{2}$$

where $w[i : j]$ represents string of symbols at positions i to j in the word w, and $nS_{ng} = |w| - N_{ng} - 1$. In most cases these values are set to $N_{ng} = 2$ and $S_{ng} = 1$; for example, partition of "world" is then "wor", "orl", "rld". Let $a \cdot b$ mean concatenation of sequences a and b, and let Σ be the alphabet. Some examples (for $N_{ng} = 2$, $S_{ng} = 1$) of word strings transformations are:

neutral transformation	$w \rightarrow w$	e.g. world → world
cyclic transformation	$w \rightarrow w{\cdot}w[0 : N_{ng} - 1]$	e.g. world → worldwo
mirror transformation	$w \rightarrow w[\lvert w\rvert - 1]{\cdot}w[\lvert w\rvert - 2]{\cdot}...{\cdot}w[0]$	e.g. world → dlrow

Let us define also functions $h(w) = w[0 : |w| - 2]$, $t(w) = w[|w| - 1]$ and function $r(i)$. When $r(i) = i$ positional ngrams are used (ngrams which „remember" their position within a sequence), and when $r(i) = 0$ free ngrams are used.

Several transformations may be applied to words simultaneously. A model of such computation is defined by several user-defined parameters: function $r(\cdot)$, ngram parameters N_{ng}, S_{ng}, composition of transformations (T), the alphabet (Σ), and the exponent W estimating importance of the function $Q'(w)$ for a given model. The total word rank function is a product over models:

$$Q(w) = \prod_{k=1}^{\#models} Q'_k(w)^{W_k} \tag{3}$$

At least one model is always required; if there are more models specified the first one is assumed to be the main model, and the remaining ones are auxiliary. Only words ranked above some threshold will be accepted as "interesting".

Function q is calculated using information from the dictionary of words and their frequencies, and optionally from a sub-dictionary containing the priming set Pr, which also contains words and their counts. Its purpose is to obtain probability-like measure of ngrams for words found in the training dictionary. For each specific model of creating words, data are put into distinct pairs of C and D tables. The steps to create these tables are presented below in the pseudo-C++ code; the calculations in lines marked with † symbol are dependent on the parameters passed to the system.

```
1. Saving data from $Dict$ to table $D_d$
   for each word w in Dict
     for each $i$-th ngram g in ng(T(w))
   †    $D_d$[r(i),h(g),t(g)] += count(w)
2. Saving data from $Pr$ to table $D_p$
   for each word w in Pr
     for each i-th ngram g in ng(T(w))
   †    $D_p$[r(i),h(g),t(g)] = 1
3. Rescaling values
   for each element e in $D_d$
   † e = $\log_{10}$(e)
4. Normalization of $D_d$, creating table $C_d$
   for b = 0 to depth($D_d$)-1, step +1
     from y = 0 to rows($D_d$)-1, step +1
   †    $C_d$[b,y] = sum($D_d$[b,y]) / sum($D_d$[b])
        s = sum($D_d$[b,y])
        for x = 0 to cols($D_d$)-1, step +1
   †       $D_d$[b,y,x] /= s
5. Creating table $C_p$
   for b = 0 to depth($D_p$)-1, step +1
     for y = 0 to rows($D_p$)-1, step +1
   †    $C_p$[b,y] = sum($D_p$[b,y]) > 0 ? 1 : 0
6. Adding table values
   † C = $C_d$ + $C_p$
   † D = $D_d$ + $C_p$
7. Normalization correction of the final tables
   for b = 0 to depth(D)-1, step +1
     for y = 0 to rows(D)-1, step +1
   †    C[b,y] /= sum(C[b])
        s = sum(D[b,y])
        for x = 0 to cols(D)-1, step +1
   †       D[b,y,x] /= s
8. Weighting the tables
   for each element e in C
   † e = $e^W$
   for each element e in D
   † e = $e^W$
```

The C and D tables serve to construct functions q and q' that may be interpreted as the probability of occurrence of a given ngram. For the first ngram h the function q may be identified as unconditional probability, and for the remaining symbols as conditional probability. In simplified notation $P(a_i) = P(a|i)$, where i is the position of a in word w over alphabet Σ. For subword v of word w let's also define:

$$P(b|a_i) = P(b_{i+|a|}|a_i); \quad h(v) = a, t(v) = b \tag{4}$$

Probability function P is calculated by a call to the table C and D: $P(a_i) = C[r(i), a]$, and $P(b|a_i) = D[r(i), a, b]$. Function $q(G)$, for a transformed sequence $G = ng(w)$, is defined as a product:

$$q(G) = P(h(G[0])_0) \prod_{i=0}^{|G|-1} P(t(G[i])|h(G[i])_i) \tag{5}$$

For the main model and positional ngrams the sequence G does not have to be treated as a continuous entity – it can be divided into independent sub-sequences in such a way that $G_0 \cdot G_1 \cdot \ldots \cdot G_J = G$. With the helper function that ranks ngrams:

$$qj(G', d) = P(h(G'[0])_d) \prod_{i=0}^{|G'|-1} P(t(G'[i])|h(G'[i])_{d+i}) \tag{6}$$

where G' is sub-sequence of G and d is initial depth of search in tables C and D, the counterpart of q for non-continuous case—function q'—is defined as:

$$q'(G) = \prod_{i=0}^{J} qj(G_i, d_i); \quad 0 \leqslant d_i \leqslant \mathrm{depth}(D) - |G_i|, \quad d_0 = 0 \tag{7}$$

Parameter J, defined by the user, controls the depth of search in matrix D for ngrams that are combined; when $J = 0$ there is no G division and $q' \equiv q$. Computationally the algorithm for functions q and q' seems to be straightforward – it is just some multiplication of values taken from the tables. However, when creating all words of a given length L the naive approach will not be acceptable due to the time complexity: for function q it is $O(|\Sigma|^L)$. To avoid lengthy computations optimized algorithm has been used, with cut-offs for words that fall below desired rank threshold at an early stage. Detailed discussion of these technical improvements are beyond the scope of this paper (Pilichowski, PhD thesis, 2009).

4 Results

The first test aims at finding a better name for Kindle electronic book reader that Amazon is advertising. Manually creating the whole priming set would be too tedious, therefore the user has to provide only the core set and the exclusion list (both in form of regular expressions). The priming set is obtained automatically from the dictionary adding the words that match regular expressions in the core set and do not match any words from the exclusion list.

The core priming set in this experiment was (each row is a concept group):

1.	acquir, collect, gather
2.	air, light$, lighter, lightest, paper, pocket, portable
3.	anyplace, anytime, anywhere, cable, detach, global, globe, go$, went, gone, going, goes, goer, journey, move, moving, network, remote, road$, roads$, travel, wire, world
4.	book, data, informati, knowledge, librar, memor, news, word$, words$
5.	comfort, easi, easy, gentl, human, natural, personal
6.	computer, electronic
7.	discover, educat, learn, read$, reads, reading, explor

The exclusion list contains: aird, airin, airs, bookie, collectic, collectiv, globali, globed, papere, papering, pocketf, travelog. The top-ranked results are below, with the number of domains that use the word, identified using Google at the end of 2008, given in the last column:

Created word	Google word count	No. domains
librazone	968	1
inforizine	–	–
librable	188	–
bookists	216	–
inforld	30	–
newsests	3	–
memorld	78	1
goinews	31	–
infooks	81,200	7
libravel	972	–
rearnews	8	–
informated	18,900,000	8
booktion	49	–
inforion	7,850	61
newravel	7	–
datnews	51,500	20
infonews	1,380,000	20
lighbooks	1	–
journics	763	1

Mambo has created several interesting new words, finding may words that are already used and have been created by people. The system has also proposed words the humans so far have used almost by accident – like “inforld”.

Another experiment was carried out to find the name for itself, i.e. the Mambo system, as it was preliminarily called. The parameters of all runs were exactly the same as above. The core priming set is presented in the table below:

1.	articula, name
2.	create, creating, creativ, generat, conceiv, build, make, construct, cook, formula, prepar, produc
3.	explor, discov, new$, newer$, newest$, newly$, imagin
4.	mean$, meanin, associat, idea$, ideas, cognitiv, think, thought, semant, connect, art$, artist, brain, mind, cogit
5.	system$, systems$, program, automat, computer, artifici
6.	wit$, wits$, witty$, smart, intell
7.	word, letter, languag

with the exclusion of one word: cookie. It is worth noting that this algorithm selected much fewer words as interesting than resulted from our previous experiments [3]; it has generated the following words:

Created word	**Google word count**	**No. domains**
semaker	903	9
braingene	45	–
assocink	3	–
thinguage	4,630	–
systemake	4	–
newthink	8,960	46
thinknew	3,300	43
assocnew	58	–
artistnew	1,590	1
semantion	693	6

Mambo system was capable of creating such words as “braingene” or “thinguage”. Fine-tuning the system (parameters, priming) may provide a lot of new words.

5 Conclusions

Inventing novel names is clearly a creative process that can be simulated using computational algorithms, and tested against human ingenuity. Many companies offer naming services, for example: Brands and Tags, Brighter Naming, Names and Brands, Named at Last, NameSharks. Models of neurocognitive processes involved in the process of creating and understanding novel words, capturing details at different levels of approximation, are worth developing. In this paper probabilistic approach has been presented. Despite its simplicity it gives quite interesting results. General neurocognitive principles of creativity are probably common to various tasks [2,3]. There is no doubt that algorithms based on neurocognitive inspirations are going to be quite useful in many interesting applications.

References

1. Sternberg, R.J. (ed.): Handbook of Human Creativity. Cambridge University Press, Cambridge (1998)
2. Duch, W.: Computational Creativity, World Congress on Computational Intelligence, Vancouver, Canada, pp. 1162–1169. IEEE Press, Los Alamitos (2006)
3. Duch, W., Pilichowski, M.: Experiments with computational creativity. Neural Information Processing – Letters and Reviews 11(4-6), 123–133 (2007)
4. McNamara, T.P.: Semantic Priming. Perspectives from Memory and Word Recognition. Psychology Press (2005)
5. Brants, T., Franz, A.: Web 1T 5-gram Version 1, http://www.ldc.upenn.edu/Catalog/
6. The Lexical Systems Group, The Specialist Lexicon, http://lexsrv3.nlm.nih.gov/LexSysGroup/Projects/lexicon/current/

7. Atkinson, K.: Spell Checker Oriented Word Lists, http://wordlist.sourceforge.net/
8. Cole, C: The Borgmann Project: Listing all the Words in English, http://pl.youtube.com/watch?v=kU_CiErwkFA
9. Kalat, J.W.: Biological psychology, 9th edn. Wadsworth Publishing (2006)
10. Miller, G.A., Fellbaum, C., Tengi, R., Wakefield, P., Langone, H., Haskell, B.R.: A lexical database for the English language, http://wordnet.princeton.edu/
11. Bowers, J.S., Davis, C.J., Hanley, D.A.: Automatic semantic activation of embedded words: Is there a "hat" in "that"? Journal of Memory and Language 52, 131–143 (2005)

Improving Personal Credit Scoring with HLVQ-C

A. Vieira[1], João Duarte[1], B. Ribeiro[2], and J.C. Neves[3]

[1] ISEP, Rua de S. Tomé, 4200 Porto, Portugal
{asv,jmmd}@isep.ipp.pt
[2] Department of Informatics Engineering, University of Coimbra, P-3030-290
Coimbra, Portugal
bribeiro@dei.uc.pt
[3] ISEG - School of Economics, Rua Miguel Lupi 20, 1249-078 Lisboa, Portugal
jcneves@iseg.utl.pt

Abstract. In this paper we study personal credit scoring using several machine learning algorithms: Multilayer Perceptron, Logistic Regression, Support Vector Machines, AddaboostM1 and Hidden Layer Learning Vector Quantization. The scoring models were tested on a large dataset from a Portuguese bank. Results are benchmarked against traditional methods under consideration for commercial applications. A measure of the usefulness of a scoring model is presented and we show that HLVQ-C is the most accurate model.

1 Introduction

Quantitative credit scoring models have been developed for the credit granting decision in order to classify applications as ‘good’ or `bad’, the latest being loosely defined as a group with a high likelihood of defaulting on the financial obligation.

It is very important to have accurate models to identify bad performers. Even a small fraction increase in credit scoring accuracy is important. Linear discriminant analysis still is the model traditionally used for credit scoring. However, with the growth of the credit industry and the large loan portfolios under management, more accurate credit scoring models are being actively investigated [1]. This effort is mainly oriented towards nonparametric statistical methods, classification trees, and neural network technology for credit scoring applications [1-5].

The purpose of this work is to investigate the accuracy of several machine learning models for the credit scoring applications and to benchmark their performance against the models currently under investigation.

The credit industry has experienced a rapid growth with significant increases in instalment credit, single-family mortgages, auto-financing, and credit card debt. Credit scoring models, i.e, rating of the client ability to pay the loans, are widely used by the financial industry to improve cashflow and credit collections. The advantages of credit scoring include reducing the cost of credit analysis, enabling faster credit decisions, closer monitoring of existing accounts, and prioritizing collections [4].

Personal credit scoring is used by banks for approval of home loans, to set credit limits on credit cards and for other personal expenses. However, with the growth in financial services there have been mounting losses from delinquent loans. For

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 97–103, 2009.

instance, the recent crises in the financial system triggered by sub-prime mortgages have caused losses of several billion dollars.

In response, many organizations in the credit industry are developing new models to support the personal credit decision. The objective of these new credit scoring models is increasing accuracy, which means more creditworthy applicants are granted credit thereby increasing profits; non-creditworthy applicants are denied credit thus decreasing losses.

The main research focuses on two areas: prediction of firm insolvency and prediction of individual credit risk. Due to the proprietary nature of credit scoring, there is few research reporting the performance of commercial credit scoring applications. Salchenberger et al. investigated the use of a multilayer perceptron neural network to predict the financial health of savings and loans [6]. The authors compare a multilayer perceptron neural network with a logistic regression model for a data set of 3429 S&L's from January 1986 to December 1987. They find that the neural network model performs as well as or better than the logistic regression model for each data set examined.

The use of decision trees and multilayer perceptrons neural network for personal credit scoring were studied by several authors. West [7] tested several neural networks architectures on two personal credit datasets, German and Australian. Results indicates that multilayer perceptron neural network and the decision tree model both have a comparable level of accuracy while being only marginally superior to tradition parametric methods.

Jensen [5] develops a multilayer perceptron neural network for credit scoring with three outcomes: obligation charged of (11.2%), obligation delinquent (9.6%), and obligation paid-of. He reports a correct classification of 76 - 80% with a false positive rate (bad credit risk classified as good credit) of 16% and a false negative rate (good credit risk classified as bad credit) of only 4%. This author concludes that the neural network has potential for credit scoring applications, but its results were obtained on only 50 examples.

The research available on predicting financial distress, whether conducted at the firm or individual level, suggests that recent non-parametric models show potential yet lack an overwhelming advantage over classical statistical techniques. Recently we have successfully applied new data mining models, like Hidden Layer Learning Vector Quantization (HLVQ-C) [8] and Support Vector Machines (SVM) [9] for bankruptcy prediction. However, the major drawback for using these models is that they are difficult to understand and the decisions cannot be explicitly discriminated.

This paper is organized as follows. Section 2 discusses the dataset used, the preprocessing of the data and feature selection. Section 3 presents the models and the usefulness measure. In Section 4 the results are discussed and finally section 5 presents the conclusions.

2 Dataset

The database contains about 400 000 entries of costumers who have solicited a personal credit to the bank. The value solicited ranges from 5 to 40 kEuros and the payment period varies between 12 to 72 months.

Table 1 presents the definitions of the eighteen attributes used by the bank. Eight of these attributes are categorical (1, 2, 3, 4, 5, 8, 9 and 10) and the remaining continuous. Most of the entries in the database have missing values for several attributes. To create a useful training set we select only entries without missing values.

The database also contains the number of days that each client is in default to the bank concerning the payment of the monthly mortgage. This quantity is usually called "days into arrears" and in most cases is zero. A client is considered delinquent when this number is greater than 30 days. We found 953 examples in the database within this category. To create a balanced dataset an equal number of randomly selected non-default examples were selected, reaching a total of 1906 training cases. We called this dataset 1.

We also created a second dataset where the definition of credit delinquency was set to 45 days into arrears. This dataset is therefore more unbalanced containing 18% of defaults and 82% non-defaults. This is called dataset 2.

Table 1. Attributes used for credit scoring. Marked bold are the selected attributes.

#	Designation	#	Designation
1	Professional activity	10	Nationality
2	Previous professional activity	**11**	Debt capacity
3	Zip code	12	Annual costs
4	Zip code – first two digits	13	Total income
5	Marital status	14	Other income
6	Age	15	Effort ratio
7	Number of dependents	16	Future effort ratio
8	Have home phone	**17**	Number of instalments
9	Residential type	**18**	Loan solicited

2.1 Feature Selection

Several feature selection algorithms were used to exclude useless attributes and reduce the complexity of the classifier. Due to the presence of many categorical attributes, feature selection is difficult. Several methods were used to test the consistency of the selection: SVM Attribute Evaluation, Chisquared and GainRatio. Each method selected slightly different sets of attributes. We choose the following set of six attributes with has the highest consensus among all rankers: 1, 3, 4, 11, 17 and 18.

3 Models Used

The data was analysed with five machine learning algorithms: Logistic, Multilayer Perceptron (MLP), Support Vector Machine (SVM), AdaBoostM1 and Hidden Layer Learning Vector Quantization (HLVQ-C).

For MLP, we used a neural network with a single hidden layer with 4 neurons. The learning rate was set to 0.3 and the momentum to 0.2. The SVM algorithm used was the LibSVM [10] library with a radial basis function as kernel with the cost parameter

C = 1 and the shrinking heuristic. For AdaBoostM1 algorithm we used a Decision Stump as weak-learner and set the number of iterations to 100. No resampling was used.

The HLVQ-C algorithm was developed to classify high dimensional data [8]. It is implemented in four steps. First, a multilayer perceptron is trained using backpropagation. Second, supervised Learning Vector Quantization is applied to the outputs of the last hidden layer to obtain the code-vectors $\vec{w}_{ci}$ corresponding to each class c_i in which data are to be classified. Each example, $\vec{x}_i$, is assigned to the class c_k having the smallest Euclidian distance to the respective code-vector:

$$k = \min_j \left\| \vec{w}_{c_j} - \vec{h}(\vec{x}) \right\| \tag{1}$$

where $\vec{h}$ is a vector containing the outputs of the hidden layer and $\|\cdot\|$ denotes the usual Euclidian distance. In the third step the MLP is retrained but with two differences regarding conventional multilayer training. First the error correction is not applied to the output layer but directly to the last hidden layer. The second difference is in the error correction backpropagated to each hidden node:

$$E_1 = \frac{1}{2}\sum_{i=1}^{N_h}\left(\vec{w}_{ck} - \vec{h}(\vec{x}_i)\right)^2 \tag{2}$$

After retraining the MLP a new set of code-vectors,

$$\vec{w}_{c_i}^{\,new} = \vec{w}_{ci} + \Delta\vec{w}_{ci} \tag{3}$$

is obtained according to the following training scheme:

$$\begin{aligned} \Delta\vec{w}_{ci} &= \alpha(n)(\vec{x} - \vec{w}_{c_i}) \quad \text{if } \vec{x} \in \text{ class } c_i\,, \\ \Delta\vec{w}_{c_i} &= 0 \qquad\qquad\qquad \text{if } \vec{x} \notin \text{class } c_i \end{aligned} \tag{4}$$

The parameter α is the learning rate, which should decrease with iteration n to guarantee convergence. Steps two and three are repeated following an iterative process. The stopping criterium is met when a minimum classification error is found.

The distance of a given example x to each prototype is:

$$d_i = \left\| \vec{h}(\vec{x}) - \vec{w}_{c_i} \right\| \tag{5}$$

which is a proximity measure to each class.

The final step of the algorithm is then applied.

Each test example, $\vec{x}^{\,i}$, is included in the training set and the neural network retrained. Since the class membership of this example is unknown, we first assign it to class “0” and determine the corresponding output $y_0(\vec{x}^{\,i}) = y_0^i$ as well as the respective distances to each class prototype,

$$\vec{d}_0^{\,i} = (d_0^{c0}, d_0^{c1}) = \left(\left\| h_0(\vec{x}^{\,i}) - w_{c0} \right\|, \left\| h_0(\vec{x}^{\,i}) - w_{c1} \right\| \right) \tag{6}$$

In a second step the network is retrained considering the example as class “1”. The new output $y_1(\vec{x}^{\,i}) = y_1^i$ and the respective distances to the prototypes are obtained in a similar way, thus:

$$\vec{d}_1^{\,i} = (d_1^{c0}, d_1^{c1}) = \left(\left\| h_1(\vec{x}^{\,i}) - w_{c0} \right\|, \left\| h_1(\vec{x}^{\,i}) - w_{c1} \right\| \right) \tag{7}$$

From these outputs, y_0^i and y_1^i, the choice is made following the heuristic rule:

$$y^i = y_0^i \text{ if } d_0^{c0} < d_0^{c1}$$
$$y^i = y_1^i \text{ if } d_1^{c1} < d_1^{c0}.$$

3.1 Usefulness of a Classifier

Accuracy is a good indicator, but not the only criteria, to choose the appropriate classifier. We introduce a measure of the usefulness of a classifier, defined by:

$$\eta = E - L, \tag{8}$$

where *E* is the earnings obtained by the use of the classifier and *L* the losses incurred due to the inevitable misclassifications.

Earning, for the bank point of view, results from refusing credit to defaults clients, and can be expressed as:

$$E = NV(1 - e_I)x \tag{9}$$

where *N* is the number of loans applicants, *V* the average value of a loan, e_I the type I error and *x* the typical percentage of defaults in the real sample. For simplicity we are assuming a Loss Given Default (LGD) of 100%.

Losses results from excluding clients that were incorrectly classified as defaults. In a simplified way they can be calculated as:

$$L = mNV(1 - x)e_{II} \tag{10}$$

where *m* is the average margin typically obtained by the bank in a loan. The net gain in using a classifier is:

$$\eta = NV\left[x(1 - e_I) - (1 - x)e_{II}m\right]. \tag{11}$$

To have $\eta > 0$ we need:

$$\frac{x}{1 - x} > mG, \tag{12}$$

where $G = \frac{e_{II}}{1-e_I}$, is a measure of the efficiency of the classifier. This quantity should be the lowest possible. Assuming x small and e_I = 0.5 , we should have $x > 2me_{II}$.

4 Results

In table 2 we compare the efficiency of the classifiers on two datasets using 10-fold cross validation. For dataset 1, most classifiers achieve a good accuracy in detecting defaults but at the cost of large type II errors. Since real data is highly unbalanced, most cases being non-defaults, this means that more than half of clients will be rejected. SVM is the most balanced classifier while HLVQ-C achieved the highest accuracy on both datasets.

Since dataset 2 is more unbalanced and the default definition more strict, error type II decreased considerably while error type I increased. More important, the usefulness of the classifier, measured by *G*, improved substantially. The HLVQ-C is again the best performer, either on accuracy and usefulness, and AdaboostM1 the second best. Logistic is the worst performer.

Following our definition, Eq. 12, for the classifier to be useful the dataset has to have about 6% defaults, considering the best model (HLVQ-C), and as much as 11% for the Logistic case (setting *m* = 0.5).

To increase the usefulness, i.e. lower *G*, error type II should decrease without deteriorating error type I. This can be done either by using a more unbalanced dataset or applying different weights for each class. The exact proportion of instances in each class in the dataset can be adjusted in order to minimize G.

Table 2. Accuracy, error types and usefulness of different models in the two datasets considered. Values are in percentage.

	Classifier	**Accuracy**	**Type I**	**Type II**	**G**
Dataset 1	Logistic	66.3	27.3	40.1	54.8
	MLP	67.5	8.1	57.1	61.1
	SVM	64.9	35.6	34.6	52.3
	AdaboostM1	69.0	12.6	49.4	55.7
	HLVQ-C	**72.6**	5.3	49.5	52.3
Dataset 2	Logistic	81.2	48.2	11.0	21.2
	MLP	82.3	57.4	9.1	20.1
	SVM	83.3	38.1	12.4	19.3
	AdaboostM1	84.1	45.7	8.0	14.7
	HLVQ-C	**86.5**	48.3	6.2	**11.9**

5 Conclusions

In this work we compared the efficiency of several machine learning algorithms for credit scoring. Feature selection was used to reduce complexity and eliminate useless attributes. From the initial set of 18 features only 6 have been selected.

While MLP slightly improves the accuracy of Logistic regression, other methods show considerable gains. AdaboostM1 boosts the accuracy by 3% and HLVQ-C up to 5%.

The price to be paid for the accurate detection of defaults is a high rate of false positives. To circumvent this situation an unbalanced dataset was used with a more strict definition of default. A measure of the usefulness of the classifier was introduced and we showed that it improves considerably on this second dataset.

Acknowledgments. The authors would like to acknowledge the financial support from Fundação da Ciência e Tecnologia, project PTDC/GES/70168/2006.

References

1. Brill, J.: The importance of credit scoring models in improving cashflow and collections. Business Credit 7, 1 (1998)
2. Tam, K.Y., Kiang, M.Y.: Managerial applications of neural networks: the case of bank failure predictions. Management Science 47, 926 (1992)
3. Davis, R.H., Edelman, D.B., Gammerman, A.J.: Machine learning algorithms for credit-card applications. IMA Journal of Mathematics Applied in Business and Industry 51, 43 (1992)
4. Desai, V.S., Crook, J.N., Overstreet, G.A.: A comparison of neural networks and linear scoring models in the credit union environment. European Journal of Operational Research 37, 24 (1996)
5. Jensen, H.L.: Using neural networks for credit scoring. Managerial Finance 26, 18 (1992)
6. Salchenberger, L.M., Cinar, E.M., Lash, N.A.: Neural networks: a new tool for predicting thrift failures. Decision Sciences 23, 899 (1992)
7. West, D.: Neural Network credit scoring models. Computers & Operations Research 27, 1131 (2000)
8. Vieira, A.S., Neves, J.C.: Improving Bankruptcy Prediction with Hidden Layer Learning Vector Quantization. Eur. Account. Rev. 15(2), 253–275 (2006)
9. Ribeiro, B., Vieira, A., Neves, J.C.: Sparse Bayesian Models: Bankruptcy-Predictors of Choice? In: Int. J. Conf. Neural Networks, Vancouver, Canada, pp. 3377–3381 (2006)
10. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines,Technical Report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan (2000)

Architecture of Behavior-Based Function Approximator for Adaptive Control

Hassab Elgawi Osman

School of Engineering, Tokyo Institute of Technology, Japan
osman@isl.titech.ac.jp

Abstract. This paper proposes the use of behavior-based control architecture and investigates on some techniques inspired by Nature- a combination of reinforcement and supervised learning algorithms to accomplish the sub-goals of a mission of building adaptive controller. The approach iteratively improves its control strategies by exploiting only relevant parts of action and is able to learn completely in on-line mode. To illustrate this, it has been applied to non-linear, non-stationary control task: Cart-Pole balancing. The results demonstrate that our hybrid approach is adaptable and can significantly improve the performance of TD methods while speed up learning process.

1 Introduction

Based on reinforcement learning (RL), agents can be partially successful to perform mode-free learning of action policies of some control problems. The biggest problem of a RL algorithm when applied to a real system is the curse of dimensionality, and the reliance on a number of assumptions about the nature of environment in which the learning takes place. e.g., it is usually assumed that the process to be controlled is either open loop stable [1] or of slow dynamic. Unfortunately, these assumptions are usually violated by any control application domain operating in the real world. Due to the above limitations in adaptability of the current RL algorithms interest to a hybrid that combines different learning strategies model arose as well. The success of RL method in application to the intelligent control of continuous control systems is turned out to be depend on the ability to combine proper function approximation method with *temporal-difference* (TD) methods such as Q-learning and value iteration [8,9,10]. This paper proposes the use of behavior-based control architecture and investigates on some techniques inspired by Nature- a combination of reinforcement and supervised learning algorithms to accomplish the sub-goals of a mission of building adaptive controller. In the presented approach, the above mentioned limitations are solved by a hybrid architecture combining temporal-difference (TD) learning algorithm with on-line variant of random forests (RF) learner that we have developed recently [5]. The variant of TD learning used here is TD(λ) [7] when $\lambda = 1$, uses the ϵ-greedy policy for action selection. We call this implementation Random-TD. The approach iteratively improves its value function by exploiting only relevant parts of action space, and is able to learn completely in the on-line mode, and since we do not freeze the learning, it can adapt to a changing situation.

[1] There is no set procedures to tune controllers.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 104–111, 2009.

Such capability of on-line adaptation would take us closer to the goal of more robust and adaptable control. Learning is expected to be faster than in TD learning, which uses stochastic search. Our results below on nonlinear and non-stationary control task: Cart-Pole balancing confirms this prediction, and its also suggested that the learning algorithm could become scalable and produce large and adaptive systems.

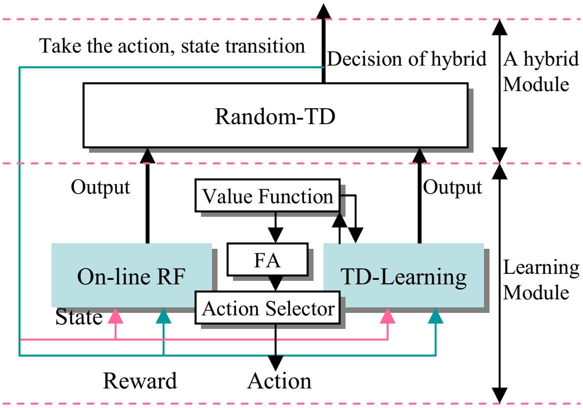


Fig. 1. Random-TD architecture

2 Random-TD Architecture

Figure 1 demonstrates the architecture of the proposed Random-TD. It consists of two modules, a learning module and a hybrid module. These two modules interact with each other. The learning module is in the first level; in which both on-line RF and TD learner learn their control policies, based on its current policy and state. Then, each learner submits its decision of the selected action or the preference of actions to the hybrid module, where Random-TD learns the combined policies. The action selector chooses an action based on the value function and some exploration and exploitation strategies. The hybrid module is at the second level, where the control policies from learning module are dynamically aggregated. After that, the hybrid module sends a final decision of action back to the learning module. Then every learner in the learning module takes the action, transits to a new state, and obtains an instant reward. In this case rewards are given for smoothing controlling the cart-pole task. Then, each learner updates its policy. Repeat for a number of steps or until several criteria are satisfied. Because both the TD-algorithm and RF run on-line, this is frees us the curse of dimensionality in a sense that memory requirements need not be exponential in the number of dimensions. The overall effect is particularly efficient computationally. The novelty of our approach stems from its orientation towards the application of knowledge based function approximation, as a result of aggregate supervisor learning with TD learning. From this approach, we hope to highlight some of the strengths of RF function approximator to the RL problems.

3 A Hybrid MDP

Let us consider a dynamic control system:

$$\frac{ds(t)}{dt} = f(s(t), (a(t)) \tag{1}$$

RL solve sequential decision making problems that can be modeled by stochastic process called *Markov Decision Processes* (MDPs). In our modeling, we assume a large but limited state space represented by control input matrix Ψ, at any given time step $t = 0, 1, 2, \cdots$, an agent perceives its *state* s_t and selects an *action* a_t. By exploring a state space an agent tries to learn the best control *policy*, $\pi : S \rightarrow A$, which maps states to actions by estimating Bellman error. The system (environment) responds by given the agent some (possibly zero) numerical *reward* $r(s_t)$ and changing into state $s_{t+1} = \delta(s_t, a_t)$. Estimate approximation for reward represented as a vector $R \in \Re^{|s|}$, are incremented by $r_t\phi_t$ and $\phi_t(\phi_t - \gamma\phi_{t+1})$, where ϕ_t, ϕ_{t+1} are control variable vectors at time step t and $t + 1$, and the transition probabilities under π can be represented as a matrix $P \in \Re^{|s|\times|s|}$. The state transition may be determined solely by the current state and the agent's action or may also involve stochastic processes. Given two Markov decision process M_1 and M_2 which share the same state space S but have two different action spaces A_1 and A_2, respectively, a new Markov decision process M_{12} can be defined by the composition of M_1 and M_2 such that at each time step the learning agent select one action from A_1 and one from A_2. The transition and cost function of the composite process M_{12} are define by

$$\delta_{12}(s, (a_1, a_2)) = \delta_2(\delta_1(s, a_1), a_2) \quad \text{and} \tag{2}$$

$$r_{12}(s, (a_1, a_2)) = r_1(s, a_1) + \gamma r_2(\delta_1(s, a_1, a_2)). \tag{3}$$

4 Random Forests in Reinforcement Learning

Learning of RF [2] is refined in this paper to deal with learning control scheme, on-line learning and other adaptability issues. Our implementation is structured by the desire to supply a behavioral guarantee based on a convergent learning algorithm.

4.1 On-Line RF

The structure of on-line forest is shown in Figure 2. The state space is partitioned onto $\Re^2$ into g disjoint group, in order to transfer a state s from the input space to a vector of input controls (actions)(See SELECT VARIABLE in Algorithm.1). The state is represented as control vectors; for each $s \in S$:

$$\phi_s = [(\phi_s(1), \phi_s(2), \cdots, \phi_s(n))]^T \tag{4}$$

$$V_t(s, \theta_t) = \theta_t^T\phi_s = \sum_{i=1}^{n} \theta_t(i)\phi_s(i) \tag{5}$$

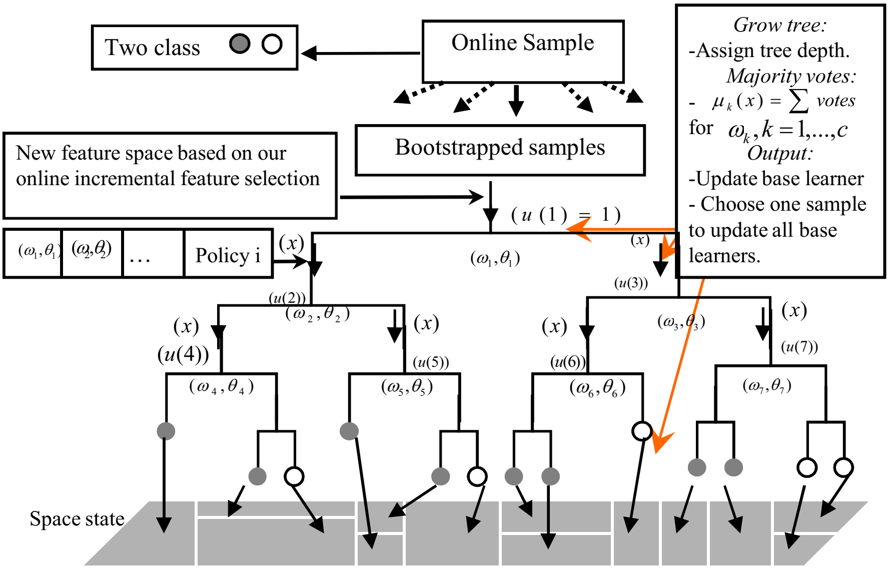


Fig. 2. A simple parameter representation of weights for a forest. The fitness of the policy is the reward “payoff” when the agent uses the corresponding on-line RF as its decision policy.

At each sub-space a local estimation is performed in order to find a good feature matrix $\varPsi$. Then value function is estimated from control (action) space, $\hat{V} = \vec{\theta}^{\mathrm{T}} \cdot \vec{A}_s$, where $\vec{\theta}$ is the parameter vector and $\vec{A}_s$ is the input controls vector. Based on control input ranking we develop a new, conditional permutation scheme for the computation of *action* importance measure [5]. According to control input ranking results, different yet random *action* subsets are used as new *action* spaces for learning a diverse base-learners (decision trees). In contrast to off-line random forests, where the root node always represents the class in on-line mode, the tree adapts the decision at each intermediate node (nonterminal) from the response of the leaf nodes, which characterized by a vector (w_i, θ_i) with $\|w_i\| = 1$. Nodes are number in the breadth-first order with the root node numbered as 1, the activation of two child nodes $2i$ and $2i+1$ of node i is given as

$$u_{2i} = u_i.f(w_i^{'}s + \theta_i) \tag{6}$$

$$u_{2i+1} = u_i.f(-w_i^{'}s + \theta_i) \tag{7}$$

where u_i represents the activation of node i, and $f(.)$ is chosen as a sigmoidal function. Consider a sigmoidal activation function $f(.)$, the sum of the activation of all leaf nodes is always unity provided that the root node has unit activation.

4.2 Learning Value Function

Individual trees in RF are incrementally generated by specifically selected subsamples from the new *action* spaces. The decision trees consist of two types of nodes: *decision nodes*, corresponding to state variables and *leaf nodes*, which correspond to all possible actions that can be taken. In a decision node a decision is taken about one of the input. Each leaf node stores the state values for the corresponding region in the state space, meaning that a leaf node stores a value for each possible action that can be taken. The tree starts out with only one leaf that represents the entire input space. So in a leaf node a decision has to be made whether the node should be split or not. Once a tree is constructed it can be used to map an input vector to a leaf node, which corresponds to a region in the state space. We will use temporal-difference learning (TD) [7] to associate a value with each region. We favoured TD learning in this implementation since it requires no model of environment, gives immediate results, and computationally cheap. In this case, the value function approximation has the form $\hat{V} = \Phi\theta$, where Φ is a $|S| \times m$ matrix m which each row contains the control variable vector for a particular state, and θ is a vector of m parameters. Typically it is desirable to have $m \ll |S|$. Usually only Φ is assumed to be given, and the learning algorithm adjusts θ.

5 Random-TD Policy Evaluation

One possible way to combine TD with RF is to choose the best combined strategy $s_i^{\rightarrow} = s_i^{\rightarrow}(s)$ given the expected combined strategy for each learners involved and to represent the value function V^{π} as combined policy of entire value function, rather than base value of a given state. This process is taken place at the hybrid module in Fig.1. Unlike in lookup tables, the value function estimate depends on a parameter vector θ_t, and only the parameter vector is updated. The Random-TD algorithm for value function approximation is shown in Algorithm 1. The function SELECT VARIABLE (s_t, e_t, A, g) uses state s_t as data points and the Bellman error estimate e_t, while the transformation from state space to feature space is done on the previous step [6]. Initially, a single feature which is 1 everywhere is defined and TD is used to compute the parameters of the approximator. On every iteration the Bellman residuals e_t are estimated for each state. A direct solution of the model parameters P and R not be feasible, either due to unknown model parameters or due to the cardinality of the state space. In either case, temporal difference (TD) methods may be employed, so the residuals can only be approximated based on an approximate model, or on sample data. The accuracy of the estimates is not crucial since on-line RF algorithm is robust to noise. In the results presented below, we use approximate Bellman errors, but using TD errors gives very similar results. Partition of state space into up to g states is done at SELECT VARIABLES stage, and it represented by combined control feature matrix Ψ. Random-TD is repeated to obtain a new approximation $\hat{V}^k$.

6 Empirical Evaluations

The results present here extends our previous results [3,4]. Before reporting our results, let us provide a brief description of the experiment.

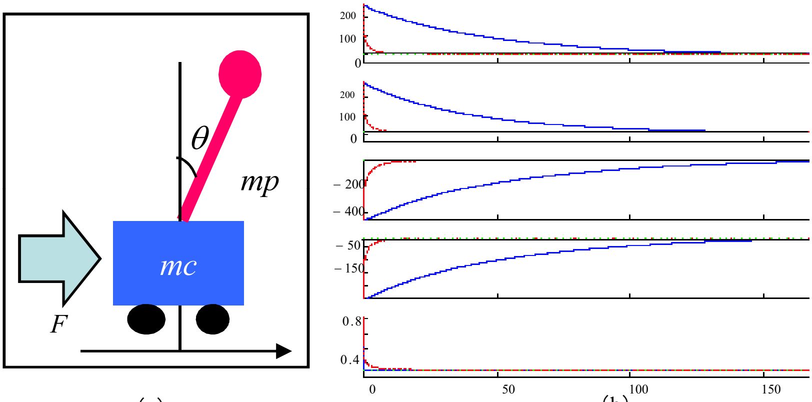
Algorithm 1. Random-TD Policy Evaluation

```
 1: Given the dimensionality $d$ to be used in each projection the number of features to be added
    in each iteration, $g$. the desire number of iteration $K$.
 2: Initialize state $s$
 3: Choose action $a$ using policy $\pi$ and observe reward $r$ and next state $\acute{s}$
 4: Update $V(s)$ such that $V(s) \leftarrow V(s) + \alpha[r + \gamma V(s) - V(s)]$ where $\alpha$ is the learning rate
    and $\gamma$ is the discount factor, $0 < \alpha < 1, 0 < \gamma < 1$
 5: $s \leftarrow \acute{s}$
 6: Repeat steps 3-5 until episode ends
 7: $\varPhi^0 \leftarrow \overrightarrow{1}$
 8: $\varPhi^0 \leftarrow TD(s_t, r_t, \varPhi^0)$
 9: for $k = 1, 2, \cdots, K$ do
10:     Estimate the Bellman residuals
11:     $e_t \approx R(s_t + \gamma \sum_{s \in S} P_{st,s} \hat{V}_s^{k-1} - \hat{V}_{st}^{k-1})$
12:     $\Re^2 \leftarrow$ Random-TD $(s_t, e_t, d)$
13:     $\varPsi \leftarrow$ SELECT VARIABLE $(s_t, e_t, A, g)$
14:     $\varPhi^k \leftarrow [\varPhi^{k-1}, \varPsi]$
15:     $\varPhi^k \leftarrow$ Random-TD$(s_t, r_t, \varPhi^k)$
16:     $V^k \leftarrow (\varPhi^k \theta^k)$
17: end for
18: return $\hat{V}^k$
```

Cart Pole Balancing One of the most well-known benchmarks in RL is the Cart-Pole balancing - the setup is shown in Figure 3 (a). In this control task, a pole with a point-mass on its upper end is mounted on a cart, and the goal is to balance a pole on top of the cart by pushing the cart left or right. Also the cart must not stray too far from its initial position. The state description to the agent consists of a four continuous state variables, the angle θ (radial) and speed of the pole $\acute{\phi} = \delta x / \delta t$ and the position x and speed $\acute{x} = \delta x / \delta t$ of the cart. Two actions were set up for the training and evaluation of the controller: RightForce (RF) (results in pushing the cart right), and LeftForce (LF) (results in pushing the cart left). The physical parameters (the mass of cart, mass and the length of the pole, and the coefficient of frictions) were the same as in [1]. An episode is a single trail to balance the pole and a non-zero reward is given at the end of the trail only. The reward is calculated as follows:

$$r_t = \begin{cases} 1 & \text{if } |\phi| < \frac{1}{15}\pi \text{ and } |\, x| < 1 \\ -1 & \text{otherwise} \end{cases}$$

This reward function gives a reward of -1 on failure and a reward of 1 on all other time steps. Failure is defined as a state where the angle of the pole is more than 12 degrees in either direction, or the cart is further than 1 meter from its initial position. When one of the conditions for failure was reached, the system was reset to $x = 0$, $\acute{x} = 0$ and $\theta = x$, where x is a random number in the interval $[-0.05, 0.05]$. This was also the initial state of the system.



(a)

(b)

Fig. 3. The performance of Random-TD in the Cart-Pole Balancing control system. In(a), The general setup of the Cart-Pole balancing problem. In (b), a sample learing. The dashed line denotes the Random-TD performance.

Table 1. The table gives the percentage of trials that ended with a solution that can balance the pole at least 100 s. Reward with a σ of 0.3. 20 simulations were performed.

	ϵ-greedy		ϵ-soft		softmax	
	γ	succ	γ	succ	γ	succ
TD	0.80	40%	0.80	60%	0.99	55%
Q-learing	0.90	15%	0.80	20%	0.00	80%
Random-TD	0.90	80%	0.95	65%	0.95	100%

6.1 Experimental Results and Discussion

In this section we demonstrate the performance of Random-TD and also compare it with other function approximation paradigms. Results for Cart Pole balancing are illustrated in Figure 3. In 3 (b), a sample run is shown: the Random-TD algorithm estimates the optimal solution with less than 8 minutes of simulated trail time. Table 1 reports result using different action selection policies; ϵ-greedy, ϵ-soft and softmax, and comparing with different RL algorithms; TD learning, Q-learning. With ϵ-greedy, most of the time the action with the highest estimated reward is chosen, called the greediest action. Every once in a while, say with a small probability ϵ, an action is selected at random. This method ensures that if enough trials are done, each action will be tried an infinite number of times, thus ensuring optimal actions are discovered. ϵ-soft very similar to ϵ-greedy. The best action is selected with probability $1 - \epsilon$ and the rest of the time a random action is chosen uniformly. On the other hand Softmax works by assigning a rank or weight to each of the actions, according to their action-value estimate. A random action is selected with regards to the weight associated with each action, meaning the worst actions are unlikely to be chosen. From the above experimental results, it can be concluded that, the Random-TD can obtain better performance than conventional TD learning algorithm.

7 Conclusion

In this paper we propose Random-TD representation as a new model for function approximation to solve problems associated with learning in large state and actions spaces. Despite the challenges when we aggregate supervised learning with TD-learning, we still reap benefits from both paradigms. From TD-learning we gain the ability to discover behavior that optimizes performance. From supervised learning we gain a flexible way to incorporate domain knowledge. Our empirical results demonstrate the feasibility and indicate strong potential for this proposed model. Our future work is to make Random-TD available as a general purpose automatic controller for self-configuring robots or mechanisms in an unknown and unstructured environment.

References

1. Barto, A., Sutton, R., Anderson, C.: Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man, and Cybernetics SMC-13, 834–846 (1983)
2. Breiman, L.: Random Forests. Machine Learning 45(1), 5–32 (2001)
3. Hassab Elgawi, O.: Architecture of Knowledge-Based Function Approximator. In: Bramer, M. (ed.) Research and Development in Intelligent Systems XXV. Proc. 28th SGAI Int'l. Conference on Artificial Intelligent (AI 2008), Springer, London (2008)
4. Hassab Elgawi, O.: A hybrid Architecture for Function Approximation. In: Proc. 6th IEEE Int'l. Conf on Industrial Informatics (INDIN 2008), pp. 1103–1108 (2008)
5. Hassab Elgawi, O.: Online Random Forests based on CorrFS and CorrBE. In: Proc.IEEE workshop on online classification, CVPR, pp. 1–7 (2008)
6. Keller, P.W., Mannor, S., Precup, D.: Automatic basis function construction for approximate dynamic programming and reinforcement learning. In: Proc. of the 23rd international conference on Machine learning, ICML, pp. 449–456 (2006)
7. Sutton, R.: Learning to predict by the method of temporal differences. Machine Learning 3(1), 9–44 (1988)
8. Sutton, R.: Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding. Advances in Neural Information Processing Systems 8, 1038–1044 (1996)
9. Sutton, R., McAllester, D., Singh, S., Mansour, Y.: Policy Gradient Methods for Reinforcement Learning with Function Approximation. Advances in Neural Information Processing Systems 12, 1057–1063 (2000)
10. Peter, S., Sutton, R., Kuhlmann, G.: Reinforcement Learning for RoboCup-Soccer Keepaway. Adaptive Behavior 13(3), 165–188 (2005)

On Efficient Content Based Information Retrieval Using SVM and Higher Order Correlation Analysis

Dimitrios Alexios Karras

Chalkis Institute of Technology, Dept. Automation and Hellenic Open University.,
Rodu 2, Ano Iliupolis, Athens 16342, Greece
dakarras@ieee.org, dakarras@teihal.gr, dakarras@usa.net

Abstract. Efficient retrieval of information with regards to its meaning and content is an important problem in data mining systems for the creation, management and querying of very large information databases existing in the World Wide Web. In this paper we deal with the main aspect of the problem of content based retrieval, namely, with the problem of document classification, outlining a novel improved and systematic approach to it's solution. We present a document classification system for non-domain specific content based on the learning and generalization capabilities mainly of SVM neural networks. The main contribution of this paper lies on the feature extraction methodology which, first, involves word semantic categories and not raw words as other rival approaches. As a consequence of coping with the problem of dimensionality reduction, the proposed approach introduces a novel higher order approach for document categorization feature extraction by considering word semantic categories higher order correlation analysis, both two and three dimensional, based on cooccurrence analysis. The suggested methodology compares favourably to widely accepted, raw word frequency based techniques in a collection of documents concerning the Dewey Decimal Classification (DDC) system. In these comparisons different Multilayer Perceptrons (MLP) algorithms as well as the Support Vector Machine (SVM), the LVQ and the conventional k-NN technique are involved. SVM models seem to outperform all other rival methods in this study.

1 Introduction

A number of statistical classification and machine learning techniques have been applied to automatic text categorisation, including domain-specific rule learning based systems [1], regression models [2], nearest neighbour classifiers [2], decision trees [3], Bayesian classifiers [3], Support Vector Machines [4] and neural networks [5]. On the other hand, several feature extraction methodologies have already been applied to this field. The basic methodology is the frequency of word occurrence technique representing a document through the vector space model (SMART) [6]. A multitude of feature extraction methods stem from the SMART representation, like word frequency weighting schemes (Boolean, tf x idf (inverse document frequency), tfc, entropy, etc.) [7] as well as feature extraction techniques involving dimensionality reduction through feature selection schemes (document frequency thresholding,

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 112–119, 2009.

information gain, hypothesis-testing, etc.) [7] and through feature combination or transformation (For instance, Latent Semantic Indexing (LSI) related methodologies) [7]. For a survey of the state-of-the-art in supervised text categorization we should point out [7].

In the herein introduced approach we present a supervised text categorization system for non-specific domain full-text documents, aimed at extracting meaning according to well defined and systematically derived subject categories used by librarians. This system is based on Supervised Neural Networks of the Multilayer Perceptron (MLP) and SVM type and a novel feature extraction scheme based on semantics processing through first, second and third order statistical procedures. The herein approach extends and improves the one presented in [12], modifying the feature extraction method presented there to introduce higher order correlations. More specifically, the proposed feature extraction technique has the following attributes.

- It involves the extraction of first, second and third order characteristics of the input documents, based on processing individual and pairs/triplets of word semantic categories. But despite [12], where word distances were examined, the methodology herein involved examines word pairs/triplets frequencies instead of distances
- Both the text categorization indexing scheme and the word semantic category extraction scheme are based on widely accepted state of the art approaches. More specifically, concerning the former, the widely adopted by librarians DDC methodology [8] is considered. Concerning the latter, the use of the WorldNet [8] as a tool for processing the documents and associating semantic categories to their words has been adopted due to its high credibility and acceptance among NLP (Natural Language Processing) researchers.

In the following section 2, the suggested supervised text categorization approach is described in detail. In section 3 the experimental study is outlined. And finally, section 4 presents the conclusions and prospects of this research effort.

2 The Proposed Neural Based Document Classification System Involving SVM and Higher Order Correlation Analysis of Semantic Categories

The suggested procedure is mainly divided into two phases. In the first phase we extract first, second and third order characteristics, that is, frequency of word semantic categories appearance (extraction of keywords) and word semantic categories affinity (extraction of pairs and triplets of words semantic categories), from a set of documents collected from the Internet and labeled using DDC methodology [8]. We then use the widely accepted NLP semantics extraction program, the thesaurus called WordNet, in order to put the words we have extracted into a fixed number of semantic categories. As illustrated in the next paragraphs, the application of WordNet results in substituting a word by the vector of its semantic categories. After this stage, we use these semantic categories vectors instead of the initial words for subsequently extracting the first, second and third order document features of the proposed supervised text categorization system. In the second phase, employing these features, SVM/MLP Neural Networks are trained so that they can classify documents into a set of DDC text categories (note that these document categories are not related in any way with the categories of the WordNet). The collection of documents used in this research effort, concerning the development of the proposed system, comes from the Internet site (link.bubll.ac.uk/isc2), which contains documents classified and accessed according to DDC (Dewey Decimal Classification). DDC is a document content classification system, internationally known and used in libraries and other organizations where a subject classification of documents is needed.

DDC defines a set of 10 main categories of documents that cover all possible subjects a document could refer to. Each one of them is divided into 10 Divisions and each division into 10 Sections. The classes of documents in DDC are defined through a systematic and objective way and not arbitrarily and subjectively. The DDC classes used in this paper are the following:
(1) Generalities,(2) Philosophy & related disciplines, (3) Religion, (4) Social sciences, (5) Language, (6) Pure sciences, (7) Technology (Applied sciences), (8) The arts, (9) Literature & rhetoric and (10 General geography, history, etc.

For each of these 10 classes a collection of 150 documents was made. The average length of these 1500 files was 256 words. Out of them, 1000 (10x100) documents (67%) comprise the training set and the rest 500 (10x50) documents (33%) comprise the test set. Following the process defined next, each document will be represented by a training (or test) pattern. So, there are 1000 training and 500 test patterns for training and testing the neural networks. We must note here that all collected documents are in English.

The first step is to remove a certain set of words, which are of less importance for meaning extraction. The set of such words excluded from further processing in this research effort are the ones presented in [12].

The second step involves creation of a vector containing every different semantic category and the frequency of its appearance in the text. That is, each word is sequentially extracted from the text and its corresponding set of n Wordnet semantic categories is found, word_i = (semantic_word_i_1, semantic_word_i_2, …. semantic_word_i_n). The element in position [i] of the semantic categories frequency vector is increased by one if the semantic category in position [i] of that vector is found in the set of semantic categories associated to that particular word and produced by WordNet. The total number of word semantic categories is 229. Therefore, each word frequency vector is transformed into word semantic categories frequency vector of 229 elements.

The third step in the suggested feature extraction methodology is the creation of the two Full Affinity Matrices of semantic categories involving their pairs and triplets triplets respectively. Based on the vector of semantic categories frequencies of appearance, prepared previously, we then, create accordingly the matrices of pairs and triplets of semantic words. Each cell of such matrices [i,j]/[i,j,k] contains the corresponding frequency of appearance within the text of the associated pair or triplet of semantic categories. In the case of these two affinity matrices, each pair/triplet of semantic categories [i,j]/[i,j,k] encountered in the text is considered. The element [i,j]/[i,j,k] of the semantic categories affinity matrix is changed if categories i,j/i,j,k are found in the current document position associated with the current pair or triplet of words.

Thus, after finishing the third step, each text is represented by:

- A 229-element WordNet categories frequency vector
- A 229x229-element semantic categories affinity matrix (Full Affinity Matrix)- – semantic categories pairs frequencies
- A 229x229x229-element semantic categories affinity matrix (Full Affinity Matrix) – semantic categories triplet frequencies.

The last step in the proposed feature extraction approach is to transform the above described affinity matrices into vectors of 6 elements by applying Cooccurrence Matrices Analysis to each one of them. The final result is that each text is represented by a vector of 229+6+6=241 elements by joining the semantic categories frequency vector and the two vectors obtained by applying the cooccurrence matrices analysis measures (of 6 dimensions each) to each of the two affinity matrices.

In this last step, some ideas taken from Texture Analysis are utilized. Texture analysis as a process is part of what is known as Image Processing. The purpose of texture analysis in image processing is to find relations between the pixel intensities that create an image. Such a relation is for example the mean and the variance of the intensity of the pixels. A common tool for texture analysis is the Cooccurrence Matrix [9]. The purpose of the cooccurrence matrix is to describe the relation between the current pixel intensity and the intensity (grey level) of the neighbouring pixels. The creation of the cooccurrence matrix is the first step of the texture analysis process. The second step is to extract a number of statistical measures from the matrix.

After cooccurrence matrices formation in image texture analysis, several measures are extracted from these cooccurrence matrices in order to best describe the texture of the source image in terms of classification accuracy. The following 6 measures in table 1 are an example of texture measures [9]. Cooccurrence matrices analysis in texture processing could be associated with word cooccurrence matrices for text understanding, in a straightforward way, where, each semantic word category (from 1 to 229) is associated with a pixel intensity. The rationale underlying such an association is explained below. It is clear that similar measures could be extracted for both word affinity matrices. Here, however, we extend, obviously the concept of two dimensional cooccurrence matrices into three dimensional ones. In every equation, f(r,c)/f(r,c,w) represents the position [r,c]/[r,c,w] of the associated matrix. Texture analysis process ends up with a small set of numbers describing an image area in terms of texture information, which results in information compression. Instead of the source image (a matrix of pixels) there is a set of numbers (the results of applying the texture measures) that represent textural information in a condensed way. This is the reason why it is herein attempted to associate texture analysis and text understanding analysis, namely, dimensionality reduction of the input vectors produced in text processing. This happens for both two and three dimensional cooccurrence analysis case as implemented through table 1. The same ideas emerging from texture analysis in image processing are applied to the document text categorization problem.

Table 1. The cooccurrence matrices analysis associated measures for the two proposed Affinity matrices of semantic categories pair and triplets frequencies. Note the differences in [12].

1. **Energy:** $\mathbf{M_1 = Sum\ Sum\ f(r,c)^2}$
2. **Entropy:** $\mathbf{M_2 = Sum\ Sum\ log(f(r,c) * f(r,c)}$
3. **Contrast:** $\mathbf{M_3 = Sum\ Sum\ (r\text{-}c)^2 * f(r,c)}$
4. **Homogeneity:** $\mathbf{M_4 = Sum\ Sum\ f(r,c)/(1+|r\text{-}c|)}$
5. **Correlation:** $\mathbf{M_5 = Sum\ Sum\ \{(r*c)\ f(r,c) - m_r\ m_c\}/s_r\ s_c}$
   where, $\mathbf{m_i}$ is the mean value and $\mathbf{s_i}$ is the variance of line (column) i.
6. **Inverse Difference Moment:** $\mathbf{M_6 = Sum\ Sum\ f(r,c)/((1+(r\text{-}c))}$
   where, the double summation symbol {Sum Sum} ranges for all rows and columns respectively.
7. **Energy:** $\mathbf{M_1 = Sum\ Sum\ Sum\ f(r,c,w)^2}$
8. **Entropy:** $\mathbf{M_2 = Sum\ Sum\ Sum\ log(f(r,c,w) * f(r,c,w)}$
9. **Contrast:** $\mathbf{M_3 = Sum\ Sum\ Sum\ (r\text{-}c)^2 * f(r,c)\ +}$
   $\mathbf{(r\text{-}w)^2 * f(r,w) + \ (c\text{-}w)^2 * f(c,w)}$
10. **Homogeneity:** $\mathbf{M_4 = Sum\ Sum\ Sum\ f(r,c)/(1+|r\text{-}c|) +}$
    $\mathbf{f(r,w)/(1+|r\text{-}w|) + f(c,w)/(1+|c\text{-}w|)}$
11. **Correlation:** $\mathbf{M_5 = Sum\ Sum\ Sum\ \{(r*c)\ f(r,c) - m_r\ m_c\}/s_r\ s_c\ +}$
    $\mathbf{\{(r*w)\ f(r,w) - m_r\ m_w\}/s_r\ s_w + \{(c*w)\ f(c,w) - m_c\ m_w\}/s_c}$
    $\mathbf{s_w}$
    where, $\mathbf{m_i}$ is the mean value and $\mathbf{s_i}$ is the variance of line (column) i.
12. **Inverse Difference Moment:** $\mathbf{M_6 = Sum\ Sum\ Sum\ f(r,c)/((1+(r\text{-}c))\ +}$
    $\mathbf{f(r,w)/((1+(r\text{-}w))\ + f(c,w)/((1+(c\text{-}w))}$
    where, the summation symbol {Sum Sum Sum} ranges for all rows in the three dimensions and columns respectively.

While cooccurrence matrices in texture analysis contain information about the correlation of neighbouring pixel intensities in an image, the proposed affinity matrices contain information about the correlations of WordNet semantic categories associated with a text. Autocorrelation matrices are involved in both definitions and therefore, both processes bear some similarities. Thus, the measures defined in table 1 are applied to the word affinity matrices too. After the application of cooccurrence based matrices analysis, there will be a total of 1500 vectors of 241 elements each. An important aspect concerning neural networks training with such vectors is the normalization of their input vectors. Normalization of the set of features presented in this paper is achieved by substituting each feature value x with $\frac{x - Min}{Max - Min}$ where Max and Min are the maximum and the minimum values of the set where this feature value belongs in, respectively.

3 Experimental Study and Results

In order to evaluate the proposed text categorization methodology an extensive experimental study has been conducted and herein is presented. Two different sets of experiments have been organized. The first one involves the collection of documents based on the DDC classification approach and the herein proposed feature extraction methodology, while the second one involves the DDC collection again but different, proposed in the literature and well established, feature extraction techniques [7]. In the second set of experiments, in order to validate our approach, we have applied a standard feature extraction method as the **tf x idf** word frequency weighting scheme [7] to the DDC collection of 1500 documents. Therefore, a set of 1500 document space vectors, different from the ones of the first set, has been obtained. The total number of different words encountered in all the 1500 documents, after removing the words specified in section 2 (words selection procedure), are 6533. Therefore, all the word frequencies based document space vectors used in this set of experiments are of 6533 dimensions.

In both sets of experiments the classifiers reported in the next paragraphs define the second stage of the text categorization system. While, in the first set they are applied to the proposed document space vectors, analyzed in section 2, in the second set of experiments they have been applied to the document space vectors derived using the **tf x idf** technique.

Concerning both sets of experiments, a cross-validation methodology [10] has been applied to all the text categorization methods herein involved, in order to ensure statistical validity of the results, due to the relatively not large number of documents included in this DDC collection. As mentioned above, the 1500 document space vectors constructed from the DDC collection are divided in the training set of patterns (1000 of them) and in the test set of patterns (500 of them). In order to apply the cross-validation methodology [10], 500 different pairs of such training and test sets have been created from the set of 1500 document space vectors by randomly assigning each of them to a training or test set.

A variety of MLP training models has been applied to both sets of experiments, including standard on-line Backpropagation, RPROP and SCG (Scaled Conjugate Gradients). Several architectures have been investigated of the types 241-x-10 and 6533-x-10 (input neurons- hidden layer neurons- output neurons). Only the best of these MLP model architectures and training parameters, out of the many ones investigated, are reported in tables 2 and 3, outlining the outcomes of this study. The MLP simulation environment is the SNNS (Stuttgart Neural Network) Simulator [11]. All training parameters shown for the Pattern Classifier training algorithms used are defined in [11]. For comparison reasons the LVQ and the k-NN classifier, which is reported to be one of the best algorithms for text categorization [7] are, also, involved in this experimental study.

Concerning the performance measures utilized in this study, mean, variance, maximum and minimum of the classification accuracy obtained by each text classifier involved are reported in tables 2 and 3. To this end, if one classifier results in G_1%, G_2%, G_3%, …., G_{500}% over all the 10 classes classification accuracies with respect to each one of the 500 different testing sets of patterns associated with the cross-validation procedure, then,

- Mean Accuracy = (G_1%+G_2%+G_3%+ ….+ G_{500}%)/500
- Accuracy Variance = VARIANCE (G_1%, G_2%, G_3%, …., G_{500}%)
- Max Accuracy = MAX (G_1%, G_2%, G_3%, …., G_{500}%)
- Min Accuracy = MIN (G_1%, G_2%, G_3%, …., G_{500}%)

where, each Gi% refers to the over all categories classification accuracy (number of correctly classified patterns/ total number of patterns) for the patterns encountered in the ith test set and not to each category classification accuracy separately. These performance measures are the usually involved statistical measures in cross-validation experiments [10]. The following tables 2 and 3 show the results obtained by conducting the above defined experimental study. Table 2 illustrates the results obtained by applying the proposed feature extraction methodology, while table 3 presents the results obtained by applying the **tf x idf** technique. These favorable text classification performance results show the validity of our approach concerning the document space vector extraction and the feasibility of the proposed solution in the real word data-mining problem under consideration.

Table 2. Text Categorization accuracy obtained involving the proposed, in section 2, document space vectors extraction methodology versus the method proposed by the author in [12]. The statistics were obtained after 500 runs of all algorithms.

Performance Measures **Text Classifier**	**Mean Accuracy (%) Proposed method**	**Accuracy Variance (%) Proposed method**	**Mean Ac-curacy (%) method in [12]**	**Accuracy Variance (%) method in [12]**
Support Vector Machine (RBF with 45 support vectors)	**77.5**	**2.2**	**76.1**	**2.4**
LVQ, 40 code book vectors	**72.1**	**2.2**	**69.8**	**2.4**
LVQ, 28 code book vectors	**72.5**	**2.3**	**70.6**	**2.5**
Vanilla BP, 241-12-10, (0.6, 0.2)	**62.3**	**2.5**	**60.8**	**2.8**
RPROP, 241-12-10, (1,100,10)	**77.2**	**3.1**	**75.5**	**3.5**
RPROP, 241-12-10, (0.1,50,4)	**75.5**	**3.1**	**74.1**	**3.1**
RPROP, 241-14-10, (0.1,50,4)	**63.9**	**2.5**	**62.3**	**2.7**
SCG, 241-15-10, (0,0,0,0)	**64.8**	**1.9**	**63.4**	**1.8**
Vanilla BP, 241-18-10, (0.6,0.2)	**62.8**	**1.1**	**61.5**	**1.1**
RPROP, 241-18-10, (1,100,10)	**73.3**	**0.9**	**72.2**	**0.8**
Vanilla BP, 241-24-10, (0.4,0.1)	**70.9**	**2.7**	**69.6**	**3.0**
K-NN (nearest neighbor)-(K= 30)	**70.4**	**2.2**	**69.2**	**2.6**

Table 3. Text Categorization accuracy obtained involving the tf x idf word frequency based document space vectors extraction methodology. The statistics were obtained after 500 runs of all the algorithms.

Performance Measures **Text Classifier**	*Mean Accura-cy (%)*	*Accuracy* Variance (%)	Max Accuracy (%)	Min Accuracy (%)
Support Vector Machine (RBF with 60 support vectors)	**69.1**	**2.4**	**69.8**	**65.2**
LVQ, 28 code book vectors	**67.7**	**2.2**	**69.1**	**64.9**
Vanilla BP (On-line BP), 6533-320-10, (0.4,0.5)	**57.3**	**4.1**	**62.4**	**55.6**
RPROP, 6533-320-10, (1,100,10)	**68.4**	**5.2**	**72.3**	**61.6**
SCG, 6533-320-10, (0,0,0,0)	**61.7**	**2.3**	**64.5**	**60.5**
K-NN (nearest neighbor)-(K= 30)	**67.6**	**2.1**	**68.9**	**64.7**

4 Conclusions and Prospects

This paper outlines an improved text categorization system for content information retrieval. The collection of documents and their content categorization is based on a widely accepted and systematic methodology, namely, the DDC system. The techniques suggested involve MLP/SVM neural networks and novel feature extraction methods based on first, second and third order characteristics of word concept frequencies and word affinities frequencies estimated by application of a widely accepted NLP thesaurus tool, namely, the WordNet and statistical techniques stemmed from Texture processing in image analysis. The promising results obtained are favourably compared to other well established text categorization document space vectors formation techniques. It is, also, shown that these results improve previous ones of the same author. Future aspects of our work include, also, the designing of a complete system based on Computational Intelligence and Artificial Intelligence Techniques for dealing with the full DDC text categorization problem, involving 1000 human understandable categories.

References

1. Cohen, W.J., Singer, Y.: Context-sensitive learning methods for text categorization. In: SIGIR 1996: Proc. 19th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, pp. 307–315 (1996)
2. Yang, Y., Pedersen, J.P.: Feature selection in statistical learning of text categorization. In: The 14th Int. Conf. on Machine Learning, pp. 412–420 (1997)
3. Lewis, D., Ringuette, M.: A comparison of two learning algorithms for text classification. In: 3rd Annual Symposium on Document Analysis and Information Retrieval, pp. 81–93 (1994)
4. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398. Springer, Heidelberg (1998)
5. Wiener, E., Pedersen, J.O., Weigend, A.S.: A neural network approach to topic spotting. In: 4th annual symposium on document analysis and Information retrieval, pp. 22–34 (1993)
6. Salton, G., McGill, M.J.: An Introduction to Modern Information Retrieval. McGraw-Hill, New York (1983)
7. Aas, K., Eikvil, L.: Text Categorisation: A Survey. Technical Report, Norwegian Computing Center, P.B. 114 Blindern, N-0314 Oslo, Norway (1999)
8. Chan, Mai, L., Comaromi, J.P., Satija, M.P.: Dewey Decimal Classification: a practical guide. Forest Press, Albany (1994)
9. Haralick, R.M., Shanmugam, K., Dinstein, I.: Textural Features for Image Classification. IEEE Trans. Syst., Man and Cybern. SMC-3(6), 610–621 (1973)
10. Haykin, S.: Neural Networks. A comprehensive foundation. Prentice-Hall, Englewood Cliffs (1999)
11. Zell, A., Mamier, G., Vogt, M., et al.: SNNS Stuttgart Neural Network Simulator, User Manual Version 4.1, Report No 6/95, University of Stuttgart, http://www-ra.informatik.uni-tuebingen.de/SNNS/UserManual/UserManual.html
12. Karras, D.A.: An Improved Text Categorization Methodology Based on Second and Third Order Probabilistic Feature Extraction and Neural Network Classifiers. In: Gabrys, B., Howlett, R.J., Jain, L.C. (eds.) KES 2006. LNCS, vol. 4251, pp. 9–20. Springer, Heidelberg (2006)

Part II

Neural Networks Learning Paradigm

A String Measure with Symbols Generation: String Self-Organizing Maps

Luis Fernando de Mingo López, Nuria Gómez Blas, and Miguel Angel Díaz

Departamento de Organización y Estructura de la Información
Escuela Universitaria de Informática
Universidad Politécnica de Madrid
lfmingo@eui.upm.es, nuria.gomez.blas@gmail.com, mdiaz@eui.upm.es

Abstract. T. Kohonen and P. Somervuo have shown that self organizing maps (*SOMs*) are not restricted to numerical data. This paper proposes a symbolic measure that is used to implement a string self organizing map based on *SOM* algorithm. Such measure between two strings is a new string. Computation over strings is performed using a priority relationship among symbols, in this case, symbolic measure is able to generate new symbols. A complementary operation is defined in order to apply such measure to *DNA* strands. Finally, an algorithm is proposed in order to be able to implement a string self organizing map. This paper discusses the possibility of defining neural networks to rely on similarity instead of distance and shows examples of such networks for symbol strings.

1 Introduction

Self Organizing maps are usually used for mapping complex, multidimensional numerical data onto a geometrical structure of lower dimensionality, like a rectangular or hexagonal two-dimensional lattice [4]. The mappings are useful for visualization of data, since they reflect the similarities and vector distribution of the data in the input space. Each node in the map has a reference vector assigned to it. Its value is a weighted average of all the input vectors that are similar to it and to the reference vectors of the nodes from its topological neighborhood. For numerical data, average and similarity are easily computed: for the average, one usually takes the arithmetical mean, and the similarity between two vectors can be defined as their inverse distance, which is most often the Euclidian one. However, for non-numerical data [6]– like symbol strings – both measures tend to be much more complicated to compute. Still, like their numerical counterparts, they rely on a distance measure. For symbol strings one can use the Levenshtein distance or feature distance.

For strings, one such measure is the *Levenshtein* distance [7], also known as edit distance, which is the minimum number of basic edit operations – insertions, deletions and replacements of a symbol – needed to transform one string into another. Edit operations can be given different costs, depending on the operation

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 123–130, 2009.

and the symbols involved. Such weighted *Levenshtein* distance can, depending on the chosen weighting, cease to be distance in the above sense of the word.

Another measure for quantifying how much two strings differ is *feature distance* [4]. Each string is assigned a collection of its substrings of a fixed length. The substrings the features are typically two or three symbols long. The feature distance is then the number of features in which two strings differ. It should be noted that this measure is not really a distance, for different strings can have a zero distance. Nevertheless, feature distance has a practical advantage over the Levenshtein by being much easier to compute.

A *similarity measure* is simpler than distance. Any function $\mathcal{S} : X2 \rightarrow R$ can be declared similarity – the question is only if it reflects the natural relationship between data. In practice, such functions are often symmetrical and assign a higher value to two identical elements than to distinct ones, but this is not required.

2 String Measure

Let V an alphabet over a set of symbols. A string x of length m belonging to an alphabet V is the sequence of symbols $a_1a_2\cdots a_m$ where the symbol $a_i \in V$ for all $1 \leq i \leq m$. The set of all strings that belong to V is denoted by V^*, the empty symbol is λ and the empty string is denoted by $\epsilon = (\lambda)^*$.

Let $\mathcal{O} : x \rightarrow n, x \in V, n \in \mathcal{N}$ a mapping that establish a priority relationship among symbols belonging to V, $u \leq v$ iff $\mathcal{O}(u) \leq \mathcal{O}(v)$. Obviously $\mathcal{O}^{-1}(\mathcal{O}(x)) = x, x \in V$ and $\mathcal{O}(\mathcal{O}^{-1}(n)) = n, n \in \mathcal{N}$, and $\mathcal{O}(\lambda) = 0, \mathcal{O}^{-1}(0) = \lambda$. This mapping can be extended over an string w in such a way that $\mathcal{O}(w) = \sum \mathcal{O}(w_i), w_i \in w$. Usually, such mapping $\mathcal{O}$ covers a range of integer numbers, that is, the output is $0 \leq i \leq k$, where $k = card(S), S \subseteq V$.

It is important to note that new symbols can be generated provided that given two symbols $a, b \in V$ $|\mathcal{O}(a) - \mathcal{O}(b)| > 1$, and there is no symbol c such that $\mathcal{O}(a) < \mathcal{O}(c) < \mathcal{O}(b)$. That is,

$$\mathcal{O}^{-1}(k) = \begin{cases} x \in V & \text{iff } \mathcal{O}(x) = k \\ s_k & \text{i.o.c.} \end{cases}, \text{ with } k \in \mathcal{N} \tag{1}$$

Symbolic measure between two strings $u, v \in V^*$, denoted by $\Delta(u, v)$, with $|u| = |v| = n$ is another string defined as:

$$\Delta(u, v) = \bigcup_{i=1}^{n} \mathcal{O}^{-1}(|\mathcal{O}(u_i) - \mathcal{O}(v_i)|), \text{ where } u_i/v_i \text{ is the } i\text{-th symbol} \in u/v \tag{2}$$

For example, let $u = (abcad), v = (abdac)$, and $\mathcal{O}$ the index of such symbol in the european alphabet, that is, $\mathcal{O}(a) = 1, \mathcal{O}(b) = 2, \mathcal{O}(c) = 3, \mathcal{O}(d) = 4$ then $\Delta(u, v) = \lambda\lambda a\lambda a$. If $u = (jonh), v = (mary)$ then $\Delta(u, v) = s_3njs_{11}$, two new symbols s_3, s_{11} are generated (that correspond to $s_3 = c$ and $s_{11} = k$, usually such correspondence is unknown).

A numeric value $\mathcal{D}$ can be define over a string w:

$$\mathcal{D}(w) = \sqrt{\sum_{i=0}^{|w|} \mathcal{O}(w_i)^2}, w_i \in w \tag{3}$$

It is clear to proof that: $\mathcal{D}(\Delta(u,v)) = \mathcal{D}(\Delta(v,u))$, $\mathcal{D}(\Delta(u,u)) = 0$, $\mathcal{D}(\Delta(u,\epsilon)) = \mathcal{D}(u)$ and $\mathcal{D}(\Delta(u,w)) \leq \mathcal{D}(\Delta(u,v)) + \mathcal{D}(\Delta(v,w))$.

Mappings $\mathcal{O}/\mathcal{D}$ also define a priority relationship among string in V^* is such a way that

$$u \leq v \text{ iff } \sqrt{\sum_{i=1}^{n=|u|} \mathcal{O}(u_i)^2} \leq \sqrt{\sum_{i=1}^{n=|v|} \mathcal{O}(v_i)^2} \tag{4}$$

$$u \leq v \text{ iff } \mathcal{D}(u) \leq \mathcal{D}(v) \tag{5}$$

In short, symbolic measure between two string u, v is obtained using $\Delta(u,v)$, see equation (3), and numeric measure is obtained using $\mathcal{D}(\Delta(u,v))$, see equation (2).

Let $x, y \in S \subseteq V$ two symbols belonging to alphabet, two symbols are complementary, denoted by $(x,y)^-$, iff $\Delta(x,y) = x$ or $\Delta(x,y) = y$. Such property can be extended over string, let $u, v \in S^* \subseteq V^*$, two strings are complementary, denoted by $(u,v)^-$, iff $\Delta(u,v) = u$ or $\Delta(u,v) = v$.

Theorem 1. *Let $u, v \in S^*$, u and $\Delta(u,v)$ are complementary iff $\mathcal{O}(u_i) >= \mathcal{O}(v_i)$ for all $1 \leq i \leq n$.*

Proof.

$$\Delta(u,v) = \bigcup_{i=1}^{n} \mathcal{O}^{-1}(|\mathcal{O}(u_i) - \mathcal{O}(v_i)|) \tag{6}$$

Hence:

$$\begin{aligned}\Delta(u,\Delta(u,v)) &= \bigcup_{i=1}^{n} \mathcal{O}^{-1}(|\mathcal{O}(u_i) - \mathcal{O}(\Delta(u_i,v_i))|) = \\ &= \bigcup_{i=1}^{n} \mathcal{O}^{-1}(|\mathcal{O}(u_i) - \mathcal{O}(\mathcal{O}^{-1}(|\mathcal{O}(u_i) - \mathcal{O}(v_i)|)|) = \\ &= \bigcup_{i=1}^{n} \mathcal{O}^{-1}(|\mathcal{O}(u_i) - (|\mathcal{O}(u_i) - \mathcal{O}(v_i)|)|) = \\ &= \bigcup_{i=1}^{n} \mathcal{O}^{-1}(\mathcal{O}(v_i)) = v\end{aligned}$$

□

Two strings $u, v \in S^*$ are *Watson-Crick* complementary (WC complementary), denoted by $(u, v)^{-WC}$, iff $(u_i, v_i)^-$ for all $1 \leq i \leq |u|$.

Theorem 2. *Let $u, v \in S^*$, if $(u, v)^-$ then $(u, v)^{-WC}$.*

Such duality in symbolic/numeric measures, see equations (2) (3), is a good mechanism in order to implement algorithms on biological DNA strands [8,3]. Like DNA or amino-acid sequences which are often subject to research in computational molecular biology. There, a different measure – similarity – is usually used. It takes into account mutability of symbols, which is determined through complex observations on many biologically close sequences. To process such sequences with neural networks, it is preferable to use a measure which is well empirically founded.

2.1 Different Length on Strings

Given two strings u, v, such that $|u| = n \geq |v| = m$, and $U(u)$ the set of all substring $w \subseteq u$ such that,

$$U(u)^m = \{w^{(j)} || w^{(j)}| = m, w = w_1 \cdots w_m, w_i = u_k, i = k + j\} \qquad \forall\, 0 \leq j \leq |u| - m$$

String measure between u, v, denoted by $\delta(u, v)$, is

$$\delta(u, v) = \{\Delta(s, v) | s \in U(u)^{|v|}, \mathcal{O}(\Delta(s, v)) \leq min_{x \in U(u)^{|v|}} \{\mathcal{O}(\Delta(x, v))\}\} \tag{7}$$

In this case, measure δ is a set of strings with the lower distance (see table below). Such distance can be read as a the set of matching strings with lower distance. This δ can be used to identify cuting points (index j) over a DNA string when applying a restriction enzyme, from a biological point of view.

$u = abcdabcdab, v = cda$

$U(u)^{\vert v \vert}$	u
a b c	
b c d	
c d a	$\mathcal{O}(\Delta(cda, v)) = 0$
d a b	
a b c	
b c d	
c d a	$\mathcal{O}(\Delta(cda, v)) = 0$
d a b	

$\delta(u, v) = \{\lambda\lambda\lambda, \lambda\lambda\lambda\}$

Let $|u| = |v|$, it is clear that $\delta(u, v) = \Delta(u, v)$ since $U(u) = u$.

3 String Self-Organizing Maps

The self-organizing map of symbol strings [4] (*SSOM* for short) doesnt differ much from ordinary numerical *SOM*. It is also a low dimensional lattice of neurons (usually two-dimensional quadratic or hexagonal lattice, sometimes one or three-dimensional), but instead of having a reference vector of input space dimensionality assigned to each node, reference strings are used. In the ordinary *SOM*, the reference vectors approximate the average of similar input vectors and input vectors similar to the reference vectors of the nodes from the topological neighborhood. In *SSOM*, the reference strings aproximate the averages of corresponding input strings [6].

A string self-organizing map of size n ($SSOM^n_{(i,j)}$ for short) is a construct $\varPhi = \{I, C, \varOmega\}$ where (i, j) are the dimensions of the competitive layer, other parameters are define as:

- $I = \{i_1, i_2, \cdots, i_n\}$ is the input nodes set,
- C is the competitive set, with $(i \times j)$ nodes,

$$C = \begin{Bmatrix} c_{11} & c_{12} & \cdots & c_{1j} \\ c_{21} & c_{22} & \cdots & c_{2j} \\ \cdots & \cdots & \cdots & \cdots \\ c_{i1} & c_{i2} & \cdots & c_{ij} \end{Bmatrix} \tag{8}$$

- and $\varOmega : n \times (i \times j) \rightarrow \omega_{n,ij}$ is a function that identifies the connection between a given input node i and a competitive node (i, j), where $\omega_{n,ij} \in U \subseteq V$.

Given a set $U \subseteq V$ and $S = \{s_1, s_2, \cdots, s_k\}$ of strings in $U^* \subseteq V^*$ in such a way that the length of every string $s_i \in S$ is $|s_i| = n$ and a priority relationship among strings in S defined using a given mapping $\mathcal{O}$. The problem consists on finding the set defined by mapping $\varOmega$ such that it minimizes the overall distance $\varDelta$ with respect the input set S.

The algorithm is based on the *SOM* algorithm, but in this case everything is symbolic.

1. *Inizialitation*: Each element $\omega_{n,ij}$ is randomly assigned a symbol in U.

2. *Feeding*: One string $s_i \in S$ in presented in the input nodes set I. Nodes in I work in a simple way, they just store information they received. Each node $i_j \in I$ stores one symbol of string s_i, that is, $i_j = (s_i)_j, 1 \leq j \leq n$.

3. *Propagation*: Information on input nodes will pass through connection till competitive layer. Nodes in competitive layer works as follows, and they will store this information:

$$c_{ij} = \bigcup_{k=1}^{n} \varDelta(i_k, \omega_{k,ij}) = \varDelta(s_i, (\omega_{1,ij}\omega_{2,ij} \cdots \omega_{n,ij})) \tag{9}$$

Such behavior is equivalent to compute distance between the input string and the connection string. This way competitive nodes calculate all distances with respect to the input string.

4. *Winning*: Nodes in C have all possible string measures, so there exists one node $c_{lm} \in C$ such that

$$\mathcal{D}(c_{lm}) \leq \mathcal{D}(c_{ij}), \forall i, j \tag{10}$$

that is, node c_{lm} has the lower distance.

5. *Learning*: Only winning node will adjust his weights (based on winner-takes-all algorithm) according to following equation:

$$\omega_{i,lm} = \mathcal{O}^{-1}(\mathcal{O}(\omega_{i,lm}) + \alpha(\mathcal{O}(i_i) - \mathcal{O}(\omega_{i,lm}))) \tag{11}$$

Some results, in literature, that could be checked with this new measure can be: for an example application of the string SOM, *Igor Fisher* generated a set of 500 strings by intorducing noise to 8 english words: always, certainly, deepest, excited, meaning, remains, safety, and touch, and initialized a quadratic map with the Sammon projection of a random sample from the set [1]. Another real world example is the mapping produced from 320 hemoglobine alpha and beta chain sequences of different species [2]. *SOM* and *LVQ* algorithms for symbol strings have been introduced by [5,6] and applied to isolated word recognition,

Table 1. Strings used in a *SSOM*, they have been modified with uniform noise

universe	networks	emulsion	elements	referred	printing	moonlike
vnfyctpb	ndwwprjt	fpwktjok	fogmeqvt	rggcopbb	spkkrhkh	lprnljjb
rlkxhpth	mctunpnu	hmtiqlnm	bjdpdqtv	rbhhrscg	rsfnwiqd	olrkjjjg
rojtdprg	kgrxlumr	gjsnrlpk	bjcpdotu	sbihoqed	osiouiqf	nmqklhkh
rliwbuth	lctzppmt	bjwksknq	emfjdkqv	pggeoudd	oufqsjpe	lqqpmflh
ulfudotb	ohuxrthu	eoumsllp	bobjhlsu	shihpseg	orlqvflg	lnlplfjb
unfwhuve	lcwvorls	dormsjml	bocngnvq	qcfcuqba	pohosgmi	noomjgkc
tklubstg	phwvqtmu	dnslufnq	dleleovv	segdqocc	mpimqiqg	nlpkkhhb
uniufuuc	phsxlshu	bkvnrkpn	ckbpgmqt	obicqoed	qplkulnf	lmolkjjd
wmhwcpqc	peqwpphq	flululnm	glhmekss	qgdftugf	rojnqlnj	jqrkniie
sogveusc	qfuyqukr	gnvmpgqo	gibnfptv	qgfetrfg	stikvlpg	lrmpkllg
sphsdotb	kgtznoiv	hmwlslrk	dnbneorq	rffeorff	moimtflg	lnmmkjkc
uqgscrrd	qctursltmin	boslqlrn	didmcmur	qdfeordg	qtkosiof	krmlnjnb
rmhsfrtd	pfrupqnp	bltiqfpl	foejbowu	tegbtuhb	osgnvimj	omnqkhng
xnkyequh	ndrymrls	elwlpflq	emfmhluv	phhgophe	osglsgpd	kmnnnflb
vniucqpf	kdsuptlp	cnwmtjqm	bkfpdotq	ubdcprbf	mrgotkqi	nmpomihf
vofsguth	nguyqslv	gmtkuhrl	fmelfosp	rhdgtpdd	qujqtlqj	kqrmokkf
smlthqrf	mcsymrlp	hmtipknl	djhpektq	pceeppff	srfowgmg	nmlpmgmb
skixbsvd	mdtyoukq	hpwirglq	gmhpcqrp	tgffuscg	pplowhqd	lmppkjie
rmfudtqf	lfsyqsmp	cjuouhqn	gkdkbptv	thcbsqce	nrklsfkh	mnqkjjlh
vkjvfttd	mcsynsns	gktmvhmq	dmbkfoqv	pgderpdb	qtgkuimi	oonpjgnf
uohvhrse	pertlqnv	cntouhqk	dihneqsr	scidosed	pqjosjkh	nqlqjflg
wphuhuqc	ngvyrpkq	eproqlnp	hlbmbqqr	rhdcusbc	mpkoqlqe	pnlplhhh
skltdrsd	kbtwntkt	bpxjtglk	bieoekus	ucfhoodc	mrgovgpi	pomknhhb
uplygpph	ndtuoojt	hnukvkpm	hlejcoqq	pgfbtobb	nuhpsgme	nmmqlgmb
snhxhtrb	qbsxopip	cpuiqgqq	fjhnblws	tecfsubg	sphkqjqe	kromjkjf

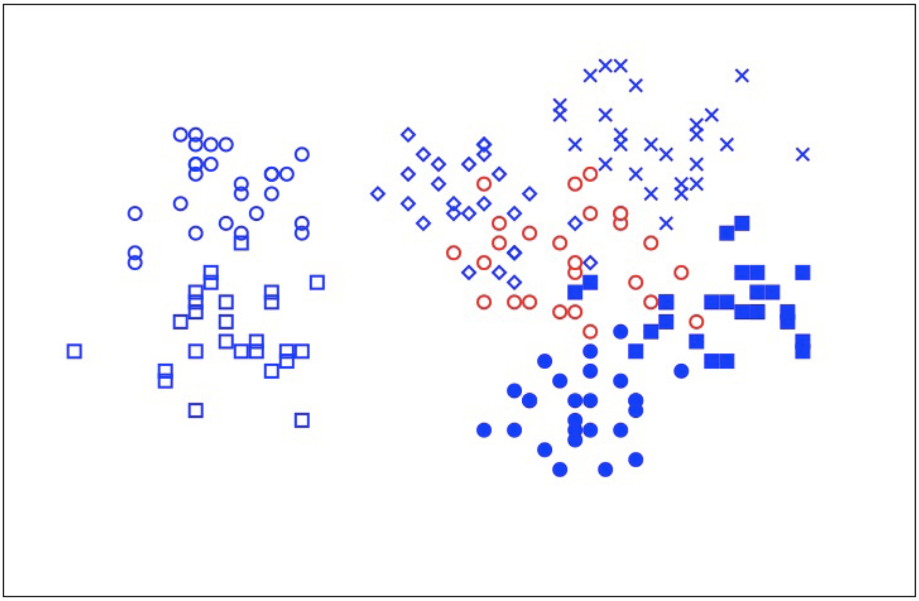


Fig. 1. Projection of string samples corresponding to table 1

for the construction of an optimal pronunciation dictionary for a given speech recognizer.

Table 1 shows strings used in a *SSOM* with a 3×3 competitive layer. Strings are obtained adding uniform noise to original strings (first row in table). After the training phase is finished clusters are named using original strings. Data in table 1 are projected in a $2-d$ surface, figure 1 shows such projection. In this case we can observe that there is a clear separation among the 7 different clusters. This is a simple projection, since it seems that some clusters are mixed, we can use the Sammon projection to obtain a better data projection.

4 Conclusions and Future Work

In some applications, like molecular biology, a similarity measure is more natural than distance and is preferred in comparing protein sequences. It is possible that such data can be successfully processed by self organizing neural networks. It can therefore be concluded that similarity-based neural networks are a promising tool for processing and analyzing non-metric data. This paper has proposed a string measure that can be applied to self organizing maps with the possibility of new symbols generation. Watson-Crick complementary concept was defined using such measure.

References

1. Fischer, I., Zell, A.: String averages and self-organizing maps for strings. In: Proceeding of the ICSC Symposia on Neural Computation (NC 2000), Berlin, Germany, May 23-26, 2000, pp. 208–215 (2000)
2. Fischer, I.: Similarity-based neural networks for applications in computational molecular biology. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 208–218. Springer, Heidelberg (2003)
3. Blas, N.G., Santos, E., Díaz, M.A.: Symbolic Learning (Clustering) over DNA Strings. WSEAS Transactions on Information Science and Applications 4(3), 617–624 (2007)
4. Kohonen, T.: Self-Organization and Associative Memory. Springer, Heidelberg (1988)
5. Kohonen, T., Somervuo, P.: Self-Organizing Maps of Symbol Strings with Application to Speech Recognition (1997)
6. Kohonen, T., Somervuo, P.: Self-organizing maps of symbol strings. Neurocomputing 21, 19–30 (1998)
7. Levenshtein, L.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics–Doklady 10, 707–710 (1966)
8. Sánchez, M.C., Gómez, N., Mingo, L.F.: DNA Simulation of Genetic Algorithms: Fitness Function. International Journal on Information Theories and Applications 14(3), 211–217 (2007)

Neural Network Smoothing of Geonavigation Data on the Basis of Multilevel Regularization Algorithm

Vladimir Vasilyev and Ildar Nugaev

Ufa State Aviation Technical University
12, K. Marx Str., 450000 Ufa, Russia
vasilyev@ugatu.ac.ru

Abstract. The problem of increasing the accuracy of geonavigation data being used for the control of the drilling oil-gas well trajectory is considered. The approach to solving the problem based on the distortion and measurement noise filtration with the use of the smoothing neural network is proposed. The generalized algorithm of the smoothing neural network design on the basis of the multilevel regularization is discussed. The peculiarities of the algorithm realization with the use of the offered vector regularization criterion of network parameters ranking is considered. The example of smoothing the geonavigation data on the basis of designed RBF network is considered.

1 Introduction

Geonavigation (geophysical navigation) is an important direction of oil-gas well drilling control development based on the continuous monitoring of the well bottom condition which is determined in its turn by the coordinates of its spatial location and the parameters of the geological environment. The continuous measurement of the azimuth and zenith angles of the drilling well trajectory in the bottom point, the natural background radiation level and the specific electric resistance of the drilling muck is carried out to monitor well bottom condition.

The peculiarity of the given parameters measurement process is the presence of distortions and measurement noises caused by intensive shock-and-vibration loads by measuring converters suffer from while drilling. Thus, the problem of increasing the geonavigation parameters measurement accuracy is one of high priority.

Practically applied approaches to solving this problem usually concern averaging multiple measurements at the separate points of the well trajectory. However, this approach becomes often inefficient because of the limited number of point measurements and the high level of obtained data dispersion.

The method of increasing the geonavigation parameters determination accuracy based on the principle of smoothing the measured data is considered bellow. The use of this principle allows us to solve the problem of the measurement errors filtration on the basis of analysis of the measurement results made sequentially by the well length.

The problem statement for smoothing the separate parameter x in this case is as follows.

For the known measurements sequence $x_m(l_i)$ containing the actual information $x(l_i)$ and the measurement errors $\varepsilon(l_i)$:

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 131–138, 2009.

$$x_m(l_i) = x(l_i) + \varepsilon(l_i),\ i=1,\ldots,\ n, \tag{1}$$

where l is the trajectory length, i – the measurement number, it is necessary to reconstruct the law of the measurement parameter $x(l)$ change in the form of the continuous smoothing model $x_s(l)$.

The corresponding values $x_s(l_i)$ calculated on the basis of the smoothing model can be used further as the required values $x_s(l)$:

$$x(l_i) \approx x_s(l_i),\ i=1,\ldots n. \tag{2}$$

One of the main difficulties arising under solving this problem is caused by the presence of an uncertain component $\varepsilon(l)$ in the initial data. The classical approach to overcome this problem is based on the use of the regularization principle taking into account some priori assumptions on the character of a desired solution [1, 2, 4, 5, 6]. The effective method of data smoothing based on the regularization principle is the construction of the smoothing model on the neural network logical basis. In a general case for this purpose one can use the Multi Layer Perceptron (MLP) or Radial Basis Function (RBF) smoothing neural network model [2].

As applied to neural networks, the regularization principle is reduced to training the neural network by optimization of the smoothing criterion representing by itself the weighted sum of two criteria [2]:

$$J=J_p+\lambda J_R(x_s(l)), \tag{3}$$

where J_p is the likelihood criterion, i.e. the sum of the squares of errors between the actual $x(l_i)$ and smoothed $x_s(l_i)$ data:

$$J_{\mathrm{P}} = \sum_{i=1}^{n} (x(l_i) - x_{\mathrm{S}}(l_i))^2; \tag{4}$$

$J_R(x_s(l)$ - the regularizing functional characterizing the variation degree of the smoothing model from a priori hypothesis, which as a rule is accepted as the hypothesis of maximal smoothness of the model $x_s(l)$. In the simplest case this criterion can be chosen as the sum of the squares of the network synaptic links;

λ - the regularization parameter determining the criteria J_p, J_R significance in (3).

The solution of the regularization task is the linear superposition of n Green's functions [1]. The special case of the Green's function is the radial basis function (RBF):

$$x_{\mathrm{S}}(l) = \frac{1}{\lambda} \sum_{i=1}^{n} w_i G(l, l_i) \tag{5}$$

where $G(l,l_i)$ is the Green's function (RBF); w_i - the weight coefficients.

It is reasonable to apply the RBF network as a solution of the regularization task. The ordinary way of RBF network learning includes finding the synaptic links weights w_i on the assumption of a priori definition of functions $G(l,l_i)$ and parameter λ.

Learning of the neural network on the basis of the regularization principle considerably increases a generalizing ability of the neural network model compared with traditional learning methods. At the same time, this approach has some disadvantages:

a) selection of the smoothing model is limited by the parameters regularization for the model from a priory chosen class without using a possibility of selecting the model's class (type of Green's functions) as itself;

b) high sensitivity of the criterion (3) to regularization parameter λ that requires the use of the higher level criteria to evaluate the relative significance of the criteria J_p, J_R. Some of the known approaches to solving this problem are: the Bayesian approach based on setting a priori distribution density for the network links weights [3], cross-validation estimation based on the result prediction [2]. These approaches are based on adequate a priori information of the being investigated process that is usually inaccessible in practice.

The approach to elimination of the mentioned disadvantages based on the proposed multilevel regularization algorithm with the use of the vector smoothing criterion is discussed below. The example of designing the neural network model providing the efficient smoothing of geonavigation data in the class of RBF networks is considered.

2 Generalized Algorithm of Smoothing Neural Network Design on the Basis of Multilevel Regularization

In order to use generalizing abilities of neural networks fully, the hierarchical procedure of its design is proposed. The initial information for this procedure includes the set of data **D** being smoothed and a priori regularizing hypothesis h on the character of the required model. The being proposed design procedure involves the following steps of the sequent restriction of the models set that can be realized on the basis of the chosen neural network type:

1) selection of the models classes set $\{M\}$;
2) selection of the models class $M\in\{M\}$;
3) selection of the model $x_S(l)\in M$;
4) testing of the model $x_S(l)$ adequacy.

In order to realize this procedure let us introduce below the tuple **P** of hierarchically ordered network parameters:

$$\mathbf{P} = (\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3), \tag{6}$$

where $\mathbf{P}_1$ are the parameters that determine the set of the models classes $\{M\}_{\mathbf{P}1}$ which can be realized by the given neural network;

$\mathbf{P}_2$ – the parameters setting the class of the models $M_{\mathbf{P}2}\in\{M\}_{\mathbf{P}1}$ realized by the given neural network and the given values of the parameters $\mathbf{P}_1$;

$\mathbf{P}_3$ – the parameters that determine the model $x_{S.}(l)\in M_{\mathbf{P}2}$ on the basis of the given neural network under the given values of the parameters $\mathbf{P}_1$ and $\mathbf{P}_2$.

Let us define the vector criterion **J** evaluating the selection of mentioned parameters **P** of the neural network for the initial data **D** and hypothesis h:

$$\mathbf{J} = \{\ J_1(\mathbf{P}_1),\ J_2(\mathbf{P}_2)|_{\mathbf{P}1}, J_3(\mathbf{P}_3)\ |_{\mathbf{P1,P2}}\,, J_4(\mathbf{D})|_{\mathbf{P1,P2,P3}}\ \}, \tag{7}$$

where $J_1(\mathbf{P}_1)$ is the criterion evaluating the selection of the parameters $\mathbf{P}_1$;

$J_2(\mathbf{P}_2)|_{\mathbf{P}1}$ – the criterion evaluating the selection of the parameters $\mathbf{P}_2$ under the given parameters $\mathbf{P}_1$;

$J_3(\mathbf{P}_3)|_{\mathbf{P}1,\mathbf{P}2}$ – the criterion evaluating the selection of the parameters $\mathbf{P}_3$ under the given parameters $\mathbf{P}_1$ and $\mathbf{P}_2$;

$J_4(\mathbf{D})|_{\mathbf{P}1,\mathbf{P}2,\mathbf{P}3}$ – the criterion evaluating the adequacy of the model $x_{S.}(l)$ under the given parameters $\mathbf{P}_1$**,** $\mathbf{P}_2$ and $\mathbf{P}_3$ with respect to the smoothed data **D.**

Unlike the classical vector criterion where partial criteria are equal by their significance and depend on one set of parameters, the vector criterion (7) uses the hierarchical principle of the partial criteria interaction based on the hierarchical ranking of the tuple of evaluated parameters (6).

As a result of the hierarchical interaction of the vector **P** and **J** components, the algorithm of the smoothing neural network design takes a form of the sequential uncertainty elimination:

Selection of the set of the smoothing models classes set $\{M\}$:

$$\mathbf{P}_{1,\min} = \arg\min_{P_1}(J_1(\mathbf{P}_1)); \tag{8}$$

2. Selection of the smoothing models class set M:

$$\mathbf{P}_{2,\min} = \arg\min_{P_2}(J_2(\mathbf{P}_2)\,|_{\mathbf{P}_{1,\min}}); \tag{9}$$

3. Selection of the smoothing model $x_S(l)\in M$:

$$\mathbf{P}_{3,\min} = \arg\min_{P_3}(J_3(\mathbf{P}_3)\,|_{\mathbf{P}_{1,\min},\mathbf{P}_{2,\min}}); \tag{10}$$

4. Testing of the model $x_S(l)$ adequacy:

a) if $J_4(\mathbf{D})|_{\mathbf{P}1,\min,\ \mathbf{P}2,\min,\mathbf{P}3,\min} \notin G$, then:

4.1. Iteration concessions Δ_1 by the model classes are carried out:

$$\mathbf{P}_{2,\ \min}^{(i+1)} = \mathbf{P}^{(i)}_{2,\ \min} \pm \Delta_1 \tag{11}$$

where i is the iteration number;

- transfer to step 3;

4.2. Iteration concessions Δ_2 by the set of the model classes are carried out:

$$\mathbf{P}_{1,\ \min}^{(i+1)} = \mathbf{P}^{(i)}_{1,\ \min} \pm \Delta_2; \tag{12}$$

- transfer to step 2;

b) if $J_4(\mathbf{D})|_{\mathbf{P}1,\min,\ \mathbf{P}2,\min,\mathbf{P}3,\min} \in G$, then:

4.3. The procedure interruption.

4.4. The adoption of the smoothing neural network parameters $\mathbf{P} = (\mathbf{P}_{1,\min}, \mathbf{P}_{2,\min}, \mathbf{P}_{3,\min})$, where G is the area of the admitted values of the model adequacy degree.

Unlike the traditional algorithm of the smoothing neural network regularization, the proposed algorithm transfers the emphasize of regularization from the stage of the model selection to the stage of the model class selection (steps 1-2). Here the model parameters can be chosen by traditional way, i.e. by training the network with use of the scalar likelihood criterion J_P in the framework of the chosen models class.

Let us consider the peculiarities of the described multilevel regularization algorithm realization on the basis of RBF network.

3 Structure of the Smoothing RBF Network

The peculiarity of RBF network is the presence of one hidden layer composed of the radial basis elements (neurons) reproducing the Gaussian response surface on the basis of RBF activation functions. RBF networks have some advantages over MLP networks. Firstly, there is no need to choose the network layers number. Secondly, the weights of the output layer neurons are being optimized with the use of effective well-known optimization methods. Thus, RBF networks have the learning rate essentially higher than for MLP networks.

The structure of the smoothing RBF network is shown in Figure 1 where S_1 is the input neuron, $S_{2,\mathrm{j}}$, $j=1,..N$ - the radial basis neurons of the hidden layer; S_3 - the output linear neuron; l_{i} - the network input data; $x_s(l_{\mathrm{i}})$ - the output smoothed data; w_j , $j=1,\dots,N$ - weights of output neuron links, -N - the number of the hidden layer neurons.

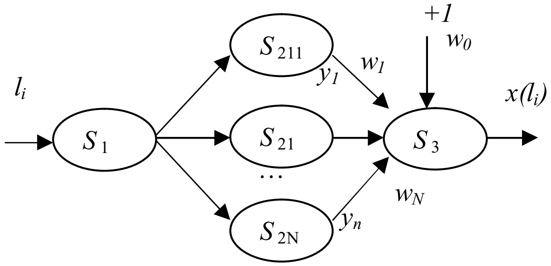


Fig. 1. Structure of smoothing RBF network

For simplicity, it is supposed here that RBF network has only one output neuron (in general case the network has several outputs). The radial-basis neuron $S_{2,j}$ structure is shown in Figure 2.

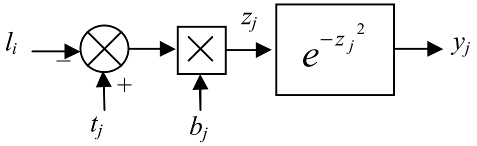


Fig. 2. Structure of the radial basis neuron $S_{2,j}$

This structure provides the calculation of the input l_i distance from the weights t_{j}, the multiplication of the result obtained by the bias b_j and the calculation of the RBF value by the formula

$$y_j = e^{-((t_j - l_i)b_j)^2} . \qquad (13)$$

The smoothing model realized on the basis of RBF network has a form:

$$x_S(l_i) = \sum_{j=1}^{N} e^{-((t_j - l_i)b_j)^2} w_j + w_0 , \tag{14}$$

where w_j , $j=0,...N$ are the weights of the output linear neuron links.

Thus, the vector of the smoothing RBF network parameters can be presented in the form:

$$\mathbf{P}_{\text{RBF}} = (N, \mathbf{B}, \mathbf{W})^T$$

where N is the number of the hidden layer neurons;

$\mathbf{B} = (b_1,..., b_N)^T$ - the vector of the hidden layer neurons' RBF biases;

W - the matrix of the synaptic links weights for the output neurons.

To realize the proposed algorithm of multilevel regularization it is necessary to define:

- the method of the RBF network parameters ranking;
- a priori hypothesis of the model regularization;
- the method of evaluating the regularization vector criterion components.

4 Method of RBF Network Parameters Ranking

The suggested method of RBF network parameters ranking is based on the analysis of their influence on the smoothing model characteristics:

1) the component $\mathbf{P}_1$ determining the models classes set $\{M\}$ can be chosen as the number of the hidden layer neurons: $\mathbf{P}_1 = N$. It means that each radial basis neuron with the parameters b_i, t_j is capable to generate a definite class of the models;

2) the vector of parameters $\mathbf{P}_2$ determining the class of the models under the given parameters $\mathbf{P}_1$ (i.e. the given number of the hidden layer neurons N) is being chosen as the vector of the hidden layer neurons' RBF biases **B**. One can see that with the purpose of algorithm simplification, the equal bias values for all N radial basis neurons can be chosen: $\mathbf{P}_2 = (b, b \ldots b)^T$**.**

The choice of parameter b value is based on the investigation results showing its significant influence on the smoothing model shape. The example of this investigation results is shown in Figure 3 where the input data of RBF network are marked by points (the geonavigation parameter θ is here the tangent inclination angle to the well trajectory in different points by its trajectory length l), the given curves correspond to the different smoothing models. These models are obtained on the basis of RBF network having 20 radial basis neurons and different values of the bias b:

- curve 1 ($b= 5.95 \cdot 10^{-5}$): the models class M is close to the linear models class;

- curve 2 ($b= 8.33 \cdot 10^{-5}$): the models class M is close to the parabolic models class;

- curve 3 ($b= 1.7 \cdot 10^{-3}$): the models class M is close to the harmonic models class;

3) as the vector of the parameters $\mathbf{P}_3$ determining the smoothing model's characteristics realized by RBF network under the given parameters $\mathbf{P}_1$ and $\mathbf{P}_2$, the matrix **W** of the output linear neuron's synaptic links weights is adopted**.**

Thus, the following relationships are distinguished:

- the models classes set $\{M\}$ is defined by the number N of the radial basis neurons;

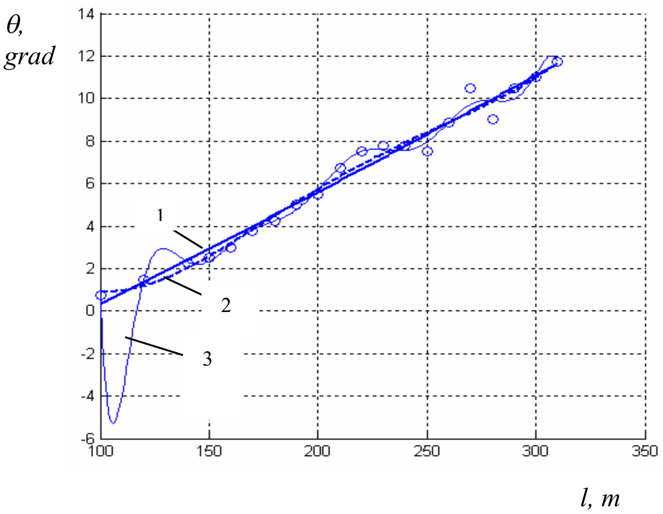


Fig. 3. Influence of bias *b* on smoothing function shape

- the class of the models *M* is defined by the bias *b* value of RBF;

- the model $x_s(l)$ is defined by the value of the links weights **W** of output linear neuron.

5 Algorithm of Smoothing RBF Network Design on the Basis of Multilevel Regularization

According to the considerations given above, the algorithm of smoothing RBF network design is as follows:

1) *selection of the smoothing models classes set {M}*: the optimal number of the radial basis neurons is determined on the basis of the regularization criterion (8):

$$\{M\} \Leftrightarrow N_{opt}\,. \tag{15}$$

The number *N* of the radial basis neurons can be chosen equal to the number of the training points in the sequence of the measured data **D.**

2) *selection of the models class M*$\in\{M\}$: the optimal value of RBF bias *b* is determined on the basis of regularization criterion (9):

$$M \Leftrightarrow b_{\text{opt}} = \arg\min_{b}(J_2(b)\,|_{N_{opt}})\,. \tag{16}$$

The solution of this task can be obtained with the use of the well known algorithms of one-parameter optimization;

3) *selection of the smoothing model* $x_S(l)\in M$: the parameters **W** are selected on the basis of RBF learning by the criterion of maximum likelihood (10) under the given parameters N_{opt}, b_{opt}:

$$x_S(l) \Leftrightarrow (\mathbf{W})_{\text{opt}} = \arg\min_{\mathbf{W}}(J_3(\mathbf{W})\,|_{N_{opt},b_{opt}})\,. \tag{17}$$

Taking into account that the model class-representatives are determined at the step 2 on the basis of the maximum likelihood criterion, the required smoothing model can be chosen as the representative of the class:

$$x_S(l) = x(l)\,|_{N_{opt}, b_{opt}, \mathbf{W}_{opt}, \mathbf{D}} \,.$$

Thus, step 3 is combined here with step 2;

4) *testing of the model $x_S(l)$ adequacy*: the information criterion $J_4(x_S(l))$ component value is checked for its belonging to accessible area G:

a) if the adequacy condition is false, i.e. $J_4 \notin G$, then a new models class is selected on the basis of concession by the criterion J_2:

$b^{(i+1)} = b^{(i)} \pm \delta$ where i is the number of the iteration step,

then transfer to step 3 to select new model parameters ($\mathbf{W}$);

b) if the adequacy condition is true, i.e. $J_4 \in G$, then the above mentioned algorithm stops, and the chosen smoothing model is adopted:

$$x_S(l) \leftrightarrow (N_{opt}, b_{opt}, \mathbf{W}_{opt}, \mathbf{A}_{opt}).$$

6 Conclusions

The algorithm of the regularization design of the smoothing neural network providing the sequential determination of the model class and its parameters is suggested. The algorithm uses more completely a neural network generalization property by introducing the regularization stage at the level of the models class regularization. The method of RBF network's parameters hierarchical ranking and their selection on the basis of the proposed regularization vector criterion is considered. The investigations carried out show the advantages of the proposed algorithm over the traditional one-level approaches in respect of increasing the smoothing model adequacy.

Acknowledgement

This research was partially supported by the Russian Foundation Basis Research (RFBR) grant № 40/51 P.

References

1. Tikhonov, A.N., Arsenin, V.A.: Solution of Ill-posed Problems. Winston & Sons, Washington (1977)
2. Haykin, S.: Neural Networks. A comprehensive foundation. Prentice Hall, New York (1994)
3. Xu, L.: Data Smoothing regularization, multi-sets-learning, and problem solving strategies. Neural Networks 16(5-6), 817–825 (2003)
4. Wahba, G.: Spline Models For Observation Data. SIAM, Philadelphia (1990)
5. Chris, M., Bishop: Training with noise is equivalent to Tikhonov regularization. Neural Computation 7(1), 108–116 (1995)
6. Burger, M., Neubauer, A.: Analysis of Tikhonov regularization for function approximation by neural networks. Neural Networks 16(1), 79–90 (2003)

Knowledge-Based Rule Extraction from Self-Organizing Maps

Chihli Hung

Department of Information Management, Chung Yuan Christian University, Taiwan
chihli@cycu.edu.tw

Abstract. The technology of artificial neural networks has been proven to be well-suited for the mining of useful information from vast quantities of data. Most work focuses on the pursuit of accurate results but neglects the reasoning process. This “black-box” feature is the main drawback of artificial neural network mining models. However, the practicability of many mining tasks relies not only on accuracy, reliability and tolerance but also on the explanatory ability. Rule extraction is a technique for extracting symbolic rules from artificial neural networks and can therefore transfer the features of artificial neural networks from “black-box” into “white-box”. This paper proposes a novel approach in which knowledge is extracted, in the forms of symbolic rules, from one-dimensional self-organizing maps. Three data sets are used in this paper. The experimental results demonstrate that this proposed approach not only equips the self-organizing map with an explanatory ability based on symbolic rules, but also provides a robust generalized ability for unseen data sets.

1 Introduction

Artificial neural networks (ANNs) have been proven to be able to deal with high dimensional data sets for several fundamental data mining tasks. However, most such work mainly concentrates on pursuing accurate results without considering the benefits from understanding and explaining the reasoning process for their mining decisions. This “black-box” feature prevents ANNs from being applied in several tasks which need stricter verification [1-2]. In other words, the practicability of a data mining technique depends not only on accuracy, reliability and tolerance for a task but also on its explanatory ability [3].

The technique which extracts symbolic rules from ANNs in order to provide a meaningful explanation is called rule extraction [4]. In this paper, we focus on extracting symbolic rules from the self-organizing map (SOM). The SOM, proposed by Teuvo Kohonen [5], is a neural topology preserving model, which projects high-dimensional data onto a low-dimensional map, usually a two-dimensional grid of units for visualization, and faithfully keeps the relationships between high-dimensional data in an output ordered map.

Most work extracts rules from a two-dimensional SOM [6-7] but such models contain a potential shortcoming in automatically having a closed boundary for each cluster. This paper projects high dimensional input samples onto a one-dimensional self-organizing map in order to extract more general rules. As the SOM is a

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 139–146, 2009.

topological ordered map, two neighbor units which have similar weights and similar input samples are mapped onto the same unit or units located nearby. Thus, a closed boundary for each cluster in the one-dimensional SOM can be found simply by evaluating the similarity of neighbor units.

On the other hand, pure unsupervised clustering methods may not be able to discern predefined classification knowledge hidden in data samples. If the results of clustering are compared with human classification knowledge, the accuracy is dependent on the difference between implicit factors of human classification labeling and explicit definitions of cluster features and similarities [8]. Thus, in this paper, we use predefined classification knowledge for finding cluster boundaries and propose a knowledge-based rule extraction from one-dimensional self-organizing maps. We use three data sets, IRIS (http://archive.ics.uci.edu/ml/), ATOM and ENGYTIME [9], to demonstrate that our proposed rule extraction model is able to extract symbolic rules from a SOM and even provides a robust generalized ability for unseen data sets.

2 Related Work

Most work on rule extraction from artificial neural networks has concentrated on supervised neural networks [4, 10-12]. By comparison with supervised neural networks, the approach of extracting rules from unsupervised neural networks, i.e. the self-organizing map (SOM), is relatively few. Darrah et al. [2] extracted rules from each best matching unit (BMU) of the trained dynamic cell structure (DCS) while the BMU is the most similar unit to the current input sample. That is, each BMU consists of one symbolic rule, which could be suitable for its associated input samples. Darrah et al. directly analyzed the maximum and minimum values for each attribute to produce an antecedent statement of a rule, i.e. the "IF" part. Antecedent statements from all attributes are joined with "AND". Conclusion statements, i.e. the "THEN" part, are also joined with "AND" while the dependent variable is continuous. However, this approach may fail to induce understandable rules when the DCS contains too many units.

Ultsch and Korus [6] proposed a rule extraction system, called REGINA, which used a unified distance matrix [13], called u-matrix method (UMM), to detect and display clusters formed by a two-dimensional trained SOM. UMM calculates the average vector of the surrounding unit vectors. Then, the trained SOM is transformed into a landscape with "hills" or "walls". All input samples that lay in a common basin belong to the same cluster as they have a stronger similarity than other input samples and thus the decision borders have been found [9]. However, the main problem of this approach is the difficulty in finding a closed and consistent decision boundary for each class, which makes it difficult for REGINA to produce the efficient symbolic rules.

Based on UMM, Malone et al. [7] built decision boundaries from a two dimensional trained SOM using the u-matrix formed by all attributes and by individual attributes respectively. That is, a data set with n attributes has n+1 u-matrices. Each individual u-matrix is then compared with the total u-matrix. An attribute is considered important if a match is found between an individual u-matrix boundary and the total u-matrix boundary. However, a closed boundary may not be found using a UMM-like approach in a two-dimensional map.

3 Finding Boundaries from a One-Dimensional Trained SOM

Extracting rules from a trained SOM can be divided into three main sub-tasks: finding a definite boundary for each cluster, selecting significant attributes for each cluster boundary and generating symbolic rules for each cluster. The main issue in the first stage is to decide the conceptual label for each SOM output unit. There is no general solution to combine neighbor units. Some researchers combine them by evaluating the mutual similarities for all SOM units based on unified Euclidean distance [9]. Some researchers take advantage of significant concepts of attributes. Two neighbor units are combined as they contain similar concepts of dominant attributes [14]. Others combine neighbor units as such units contain the same or similar concepts of dominant labels [15]. In this paper, we treat pre-labeled information as useful knowledge, so neighbor units with the same dominant label are combined in order to form a cluster.

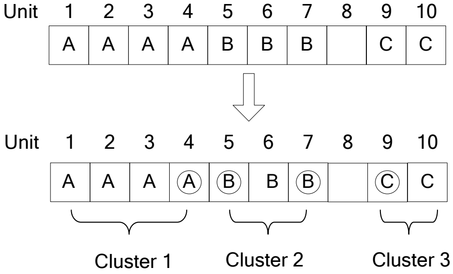


Fig. 1. A one-dimensional SOM trained map using ten output units

For example, we have a one-dimensional trained SOM which contains 10 output units as shown in Fig. 1. We assign the dominant label as the unit label. We assume that there are 10 input samples which are mapped onto Unit 1. Among these 10 input samples, seven input samples are pre-labeled as Class A, two input samples are pre-labeled as Class B and one input sample is pre-labeled as Class C. This unit is then labeled by A as it is the dominant label in this unit. We omit the dead unit, which has no associated input samples. The neighbor units with the same dominant label form a cluster. Thus, Unit 8 is a dead unit, which has no label on it. Units 1-4, Units 5-7 and Units 9-10 form different clusters with different conceptual labels, i.e. 1, 2, and 3 respectively. Neighbor units with different dominant labels are boundary units, i.e. Units 4, 5, 7 and 9, which are circled (Fig. 1).

4 Selecting Significant Attributes

More attributes mean a rule contains more logical conditions in the IF-part, but also makes it more complicated to understand. Furthermore, more attributes may lead to a more specific model, which may also cause the model to suffer from over-training. Thus, it is necessary to select significant attributes while extracting rules from a trained SOM.

U-matrix [13] is a Euclidean-distance based approach to the evaluation of the similarity of neighbor units. The Euclidean distance between Unit *i-1* and Unit *i* is shown in (1). *A* indicates the total number of attributes and *u(a)* indicates the value of Attribute *a* of *U*. In this paper, we also follow this approach and extend the original u-matrix approach to the attribute level. Thus, for each attribute, u-matrix can also be used for the evaluation of the similarity of neighbor units. We show the equation of u-matrix in the attribute level. The distance between Unit *i-1* and Unit *i* for the specific attribute *a* is shown in (2). We compare the difference of u-matrix between two levels for the evaluation of the significance of each attribute. The basic concept is that one attribute is more important than other attributes if its normalized u-matrix is more similar to its normalized unit u-matrix. We calculate u-matrix for each unit and attribute as (3) and (4) respectively. $D_{all}(U_{i-1}, U_i)$ is the Euclidean distance between the Units *i-1* and *i*. $D_{all}(U_i)$ is the average u-matrix of U_i*'s* surrounding units, i.e. $D_{all}(U_{i-1}, U_i)$ and $D_{all}(U_i, U_{i+1})$. $D_a(U_{i-1}, U_i)$ is the distance between the Units *i-1* and *i*. $D_a(U_i)$ is the average Attribute *a* u-matrix of U_i*'s* surrounding units, i.e. $D_a(U_{i-1}, U_i)$ and $D_a(U_i, U_{i+1})$. We then normalize all the weight values of u-matrix to be between 0 and 1.

$$D_{all}(U_{i-1}, U_i) = \sqrt{\sum_{a=1}^{A} (u(a)_{i-1} - u(a)_i)^2} \ . \tag{1}$$

$$D_a(U_{i-1}, U_i) = \left| u(a)_{i-1} - u(a)_i \right| \ . \tag{2}$$

$$u\text{-}matrix(attribute{=}all)=[D_{all}(U_1), D_{all}(U_1, U_2), D_{all}(U_2), D_{all}(U_2, U_3), \ldots, D_{all}(U_{i-2}, U_{i-1}), D_{all}(U_{i-1}), D_{all}(U_{i-1}, U_i), D_{all}(U_i) \ldots, D_{all}(U_{M-2}, U_{M-1}), D_{all}(U_{M-1}), D_{all}(U_{M-1}, U_M), D_{all}(U_M)] \ . \tag{3}$$

$$u\text{-}matrix(attribute{=}a)=[D_a(U_1), D_a(U_1, U_2), D_a(U_2), D_a(U_2, U_3), \ldots, D_a(U_{i-2}, U_{i-1}), D_a(U_{i-1}), D_a(U_{i-1}, U_i), D_a(U_i)\ldots, D_a(U_{M-2}, U_{M-1}), D_a(U_{M-1}), D_a(U_{M-1}, U_M), D_a(U_M)] \ . \tag{4}$$

We only keep $D_{all}(U_i)$ and $D_a(U_i)$, where *i* is from *1* to *M*. Thus, *u-matrix(attribute=all)* and *u-matrix(attribute=a)* are equal to $[D_{all}(U_1), D_{all}(U_2), \ldots, D_{all}(U_{i-1}), D_{all}(U_i), \ldots, D_{all}(U_{M-1}), D_{all}(U_M)]$ and $[D_a(U_1), D_a(U_2), \ldots, D_a(U_{i-1}), D_a(U_i), \ldots, D_a(U_{M-1}), D_a(U_M)]$ respectively. We use *u-difference* to evaluate the relationship between each attribute u-matrix and its associated unit u-matrix as shown in (5).

$$u\text{-}difference_i(attribute{=}a)=(D_a(U_i)-D_{all}(U_i))^2 \ . \tag{5}$$

To detect the significance of an attribute in a cluster, we accumulate each u-difference of units in this cluster as *acc-difference* in (6). *c* indicates a specific cluster and Units between *i* and *s* belong to cluster *c*. Then the relative difference of each attribute will be evaluated as *rel-difference* in (7). *A* is the total number of attributes and *c* is a specific cluster. A larger value of *rel-difference* for an attribute in a cluster means that this attribute is less similar to the value of unit u-matrix. Thus, this attribute is less important to describe its associated cluster. We arrange all attributes in their associated clusters in an increasing order based on their *rel-difference*. The attributes with the smallest *rel-difference* value in the ordered sequence are accumulated if the cumulative percentage equals or is less than a given threshold.

$$acc\text{–}difference_c(attribute = a) = \sqrt{\sum_{i=1}^{s} u\text{–}difference_a} \cdot \tag{6}$$

$$rel\text{–}difference_c(attribute = a) = \frac{acc\text{–}difference_c(attribute = a)}{\sum_{a=1}^{A} acc\text{–}difference_c(attribute = a)} \cdot \tag{7}$$

5 Generating Symbolic Rules

For clusters located at two ends of a one-dimensional SOM map, i.e. Clusters 1 and 3 in Fig. 2, each cluster has one boundary unit, i.e. Units 9 and 4 respectively, and one neighbor cluster, i.e. Cluster 2. Thus, we only compare each boundary unit with its neighbor boundary unit, i.e. Units 7 and 5, respectively. On the other hand, for clusters located between these two ends of a one-dimensional SOM map, there are two boundary units and two associated neighbor clusters. For example, Cluster 2 is located between Clusters 1 and 3 so we need to compare its boundary units, i.e. Units 7 and 5, with their associated boundary units, i.e. Units 9 and 4.

A cluster with a greater number of significant attributes means that this cluster is more specific. Thus, we rank symbolic rules based on the number of significant attributes. For example, the rule for Cluster 1 is ranked above Cluster 2 if Cluster 1 has more significant attributes than Cluster 2. If this condition is tied, we then rank symbolic rules based on the cumulative value of *rel-difference*. A cluster with a smaller cumulative *rel-difference* is ranked higher.

In order to make our rule extraction model more general, we consider the purity of boundary units to fine tune the decision border of neighbor clusters. The purity of boundary is defined as (8) and the decision border is defined as (9). N_l indicates the total number of samples with the dominant Label l which are projected to Unit u. N_u indicates the total number of samples which are projected to Unit u. $B_{i,j}$ is the decision border between boundary Units i and j, P is the purity of the unit and U is the weight value of unit. Originally the decision border is located between two boundary units as in Fig. 2. This decision border is then adjusted to move towards one side when the purity of the boundary unit on this side is purer than the other.

$$P(u) = \frac{N_l}{N_u} \cdot \tag{8}$$

$$B_{i,j} = U_i - \frac{(U_i - U_j) \times P(i)}{P(i) + P(j)} \cdot \tag{9}$$

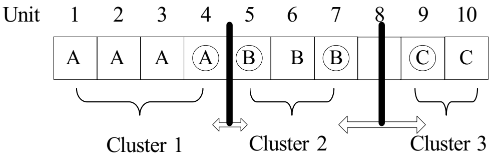


Fig. 2. Decision borders for a one-dimensional SOM trained map using ten output units

6 Experimental Results

In this paper we have used three data sets, i.e. IRIS, ATOM and ENGYTIME. The IRIS data set is made of 150 4D vectors from three classes, i.e. setosa, versicolor and virginica, each of which contains 50 input samples. We use 120 input samples as the training set and use the remaining 30 samples as the test set. The ATOM data set contains 800 input samples, which are pre-classified into two classes. We use 600 input samples as the training set and use 200 input samples as the test set. The ENGYTIME data set contains 4097 input samples, which are also pre-classified into two classes. We use 3072 input samples as the training set and use 1025 samples as the test set.

For the IRIS data set, we set a threshold, i.e. 10%, for the cumulative percentage of relative difference and have rules for the training set as shown in Fig. 3. We compare our results with the work of Malone et al. [7] and Darrah et al. [2]. The accuracy of the training set in our model is 92.50% and this accuracy is even higher for the test set, which reaches 100% and is better than other rule extraction models (Table 1).

Rule 1 : if X(a=3)< 2.5238 and X(a=4)< 0.659 then class 1 (setosa)

Rule 2 : if X(a=4)< 1.5798 and X(a=4)> 0.659 then class 2 (versicolor)

Rule 3 : if X(a=3) > 4.729 then class 3 (virginica)

Otherwise : unknown

Fig. 3. Rules extracted from a one-dimensional trained SOM with a threshold of 10% for the IRIS training set

Table 1. The performance comparison for IRIS evaluated by accuracy

	Proposed model	Malone et al. 2006	Darrah et al. 2004
Training set	92.50%	95.30%	82.00%
Test set	100.00%		

We use 25% of the threshold for the ATOM data set and 35% of the threshold for the ENGYTIME data set. Their associated rules are shown in Fig. 4 and 5. The performance for the ATOM and ENGYTIME data sets is shown in Table 2. All test sets are able to achieve a higher accuracy than their associated training sets, which explains the generality of our rule extraction model.

Rule 1: if X(*a*=3) < -2.7458 then class 1
Rule 2: if X(*a*=1) < 11.17065 and X(*a*=1) > -5.1146 then class 2
Rule 3: if X(*a*=1) > -5.1146 then class 1
Otherwise : unknown

Fig. 4. Rules extracted from a one-dimensional trained SOM with a threshold of 25% for the ATOM training set

Rule 1: if X(*a*=2) > 1.0738 then class 1
Rule 2: if X(*a*=1) < 2.7298 then class 2
Otherwise: unknown

Fig. 5. Rules extracted from a one-dimensional trained SOM with a threshold of 35% for the ENGYTIME training set

Table 2. The performance for the ATOM and ENGYTIME data sets evaluated by accuracy

Dataset	ATOM	ENGYTIME
Training set	83.67%	81.35%
Test set	85.00%	82.23%

7 Conclusions

The technique of rule extraction from artificial neural networks is a way to improve their practicability in a real-world environment. Extracting symbolic rules from SOMs could be treated as a complementary approach to the original data visualization technique. In this paper, we have proposed a novel approach of extracting rules from one-dimensional trained self-organizing maps. Most related work extracts rules from a two-dimensional SOM but suffers from automatically composing a closed boundary for each cluster. This paper projects high dimensional input samples directly onto a one-dimensional self-organizing map, so the formation of a closed boundary for each cluster is straightforward. This paper also integrates unsupervised clustering techniques with supervised classification knowledge in order to investigate human classification information hidden in data. Three data sets are used in this paper. The experimental results demonstrate that this proposed approach not only equips the self-organizing map with an explanatory ability based on symbolic rules, but also provides a robust generalized ability for unseen data sets.

References

1. Johansson, U., Niklasson, L.: Neural Networks - from Prediction to Explanation. In: Proceedings of 7th International Conference on Artificial Intelligence and Applications, Malaga, Spain, pp. 93–98 (2002)
2. Darrah, M., Taylor, B., Skias, S.: Rule Extraction from Dynamic Cell Structure Neural Networks Used in a Safety Critical Application. In: Proceedings of the 17th International FLAIRS Conference, Miami, Florida, USA (2004)

3. Michalski, R.S.: A Theory and Methodology of Inductive Learning. Artificial Intelligence 20, 111–161 (1983)
4. Andrews, R., Diederich, J., Tickle, A.: A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. Knowledge Based Systems 8, 373–389 (1995)
5. Kohonen, T.: Self-Organized Formation of Topologically Correct Feature Maps. Biological Cybernetics 43, 59–69 (1982)
6. Ultsch, A., Korus, D.: Automatic Acquisition of Symbolic Knowledge from Subsymbolic Neural Networks. In: Proceedings of the 3rd European Congress on Intelligent Techniques and Soft Computing EUFIT 1995, Aachen, Germany, August 28-31, 1995, pp. 326–331 (1995)
7. Malone, J., McGarry, K., Wermter, S., Bowerman, C.: Data Mining Using Rule Extraction from Kohonen Self-Organising Maps. Neural Computing & Applications 15(1), 9–17 (2006)
8. Hung, C., Wermter, S., Smith, P.: Hybrid Neural Document Clustering Using Guided Selforganisation and Wordnet. IEEE-Intelligent Systems 19(2), 68–77 (2004)
9. Ultsch, A.: Clustering with SOM: U*C. In: Proceedings Workshop on Self-Organizing Maps (WSOM 2005), Paris, France, pp. 75–82 (2005)
10. Núñez, H., Angulo, C., Català, A.: Rule-Based Learning Systems for Support Vector Machines. Neural Processing Letters 24(1), 1–18 (2006)
11. Saad, E.W., Wunsch, D.C.: Neural Network Explanation Using Inversion. Neural Network 20(1), 78–93 (2007)
12. Quteishat, A., Lim, C.-P.: A Modified Fuzzy Min-Max Neural Network with Rule Extraction and Its Application to Fault Detection and Classification. Applied Soft Computing 8(2), 985–995 (2008)
13. Ultsch, A., Siemon, H.P.: Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis. In: Proceedings of International Neural Networks Conference (1990)
14. Chen, H., Schuffels, C., Orwig, R.: Internet Categorization and Search: a Self-Organizing Approach. Journal of Visual Communication and Image Representation 7(1), 88–102 (1996)
15. Ritter, H., Kohonen, T.: Self-Organizing Semantic Maps. Biological Cybernetics 61, 241–254 (1989)

A Bayesian Local Linear Wavelet Neural Network

Kunikazu Kobayashi, Masanao Obayashi, and Takashi Kuremoto

Yamaguchi University, 2-16-1, Tokiwadai, Ube, Yamaguchi 755-8611, Japan
{koba,m.obayas,wu}@yamaguchi-u.ac.jp
http://www.nn.csse.yamaguchi-u.ac.jp/k/

Abstract. In general, wavelet neural networks have a problem on the curse of dimensionality, i.e. the number of hidden units to be required are exponentially rose with increasing an input dimension. To solve the above problem, a wavelet neural network incorporating a local linear model has already been proposed. On their network design, however, the number of hidden units is empirically determined and fixed during learning. In the present paper, a design method based on Bayesian method is proposed for the local linear wavelet neural network. The performance of the proposed method is evaluated through computer simulation.

1 Introduction

Wavelet neural networks (WNNs) have been mainly developed on signal processing and image processing [1]. The greatest characteristic of the WNNs is based on a temporally and spacially localized basis function, called mother wavelet. The localized basis function can identify the localization with arbitrary precision because both location and resolution can be freely adjusted by translation and dilation parameters. Radial basis function (RBF) networks also use a localized function [2], but it is difficult to adjust resolution with arbitrary precision.

Y. C. Pati et al. proposed a WNN introducing a discrete affine wavelet transform and presented the solutions to three problems on general feedforward neural networks, i.e. (1) how to determine the number of hidden units, (2) how to utilize information on training data and (3) how to escape local minima [3]. Q. Zhang et al. derived a WNN from a wavelet series expansion and mathematically proved that it can approximate any continuous functions [4]. K. Kobayashi et al. proposed a network design method utilizing a wavelet spectrum on training data [5] and another design method using genetic algorithm [6].

However, WNNs still have a problem on the curse of dimensionality, i.e. the number of hidden units rises exponentially as the number of input dimensions increases. To solve the above problem, T. Wang et al. introduced a local linear model into a WNN and developed a local linear wavelet neural network (LLWNN) [7]. After that, a lot of effort were mainly devoted into the LLWNN from parameter learning aspect. T. Wang et al. proposed two methods employing gradient descent method and genetic programing for parameter learning [7,8].

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 147–154, 2009.

Y. Chen et al. also presented two methods using gradient descent method and PSO (particle swarm optimization) for parameter learning [9,10] Most research, however, does not focused on the network design.

On the other hand, Bayesian method based on Bayesian statistics has applied to the network design of neural networks [11]. S. Albrecht et al. formulated a RBF network by Bayesian method and employed EM (Expectation-Maximization) algorithm for parameter learning [12]. C. Andrieu et al. also formulated a RBF network and approximated the integral calculation in posteriori distribution by MCMC (Markov Chain Monte Carlo) method [13]. N. Ueda et al. proposed a split and merge EM algorithm to network design and parameter learning for a mixture of experts model [14]. M. Sato et al. presented a design method based on variational Bayes method for a normalized Gaussian network (NGnet) [15]. C. C. Holmes et al. proposed a Bayesian design method for a feedforward neural network [16].

In the present paper, a Bayesian-based method for both network design and parameter learning of LLWNN is proposed. The proposed method basically follows a framework which proposed by C. C. Holmes et al. and tries to formulate a LLWNN. The LLWNN applying the proposed method is called a BLLWNN (Bayesian local linear wavelet neural network). Through computer simulation using function approximation problem, the performance of the BLLWNN is verified.

2 Local Linear Wavelet Neural Network (LLWNN)

Figure 1 shows a three-layered feedforward local linear wavelet neural network with N inputs and one output [7]. When an input vector $\mathbf{x} = \{x_1,\ x_2, \cdots, x_N\} \in R^N$ is given, LLWNN converts it to a weighted output $\hat{f}(\mathbf{x})$ with a local linear parameter c_k. That is, the output $\hat{f}$ of LLWNN is denoted by the follwoing equation.

$$\hat{f}(\mathbf{x}) = \sum_{k=1}^{K} c_k \,\Psi_k(\mathbf{x}) = \sum_{k=1}^{K} (w_{k0} + w_{k1}x_1 + \cdots + w_{kN}x_N)\,\Psi_k(\mathbf{x}). \tag{1}$$

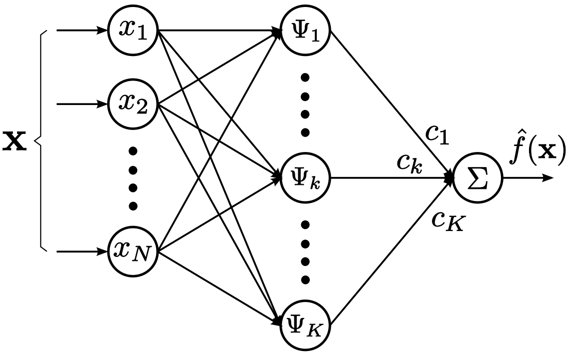


Fig. 1. Architecture of a local linear wavelet neural network

In (1), the basis function Ψ_k is defined by

$$\Psi_k(\mathbf{x}) = |\mathbf{a}_k|^{-\frac{1}{2}} \psi\left(\frac{\mathbf{x} - \mathbf{b}_k}{\mathbf{a}_k}\right), \tag{2}$$

where $\mathbf{a}_k \in R^N$ and $\mathbf{b}_k \in R^N$ refer to dilation and translation parameters, respectively. Then, the basis function ψ in (2) must satisfy the following admissible condition.

$$\int_R \frac{|\hat{\Psi}(\lambda)|^2}{\lambda} \, d\lambda < \infty, \tag{3}$$

where $\hat{\Psi}(\lambda)$ means Fourier transform of $\Psi(\mathbf{x})$. For a localized basis function, (3) is equivalent to $\int_R \psi(x)dx = 0$. That is, the basis function has no direct current component.

The difference between general WNN and LLWNN depends on the parameter c_k. The parameter c_k is scalar for WNN and a local linear model for LLWNN. This allows LLWNN could cover larger area in input space compared with WNN. As a result, LLWNN may resolve the curse of dimensionality because it can reduce the numbers of hidden units and free parameters. Furthermore, it is clarified that the local linear model realizes good interpolation even if the number of learning samples is small [7].

On network design of LLWNN, however, almost all the models empirically determine the number of hidden units [7,8,9,10]. Namely, the network structure is determined in advance and fixed during learning. Y. Chen et al. has tried to construct a LLWNN using eCGP (Extended Compact Genetic Programming) [17]. However, since they represented the network as a hierarchical structure, it seems that the numbers of hidden units and free parameters tend to be large [10].

In the present paper, a Bayesian method is applied to network design and parameter learning of LLWNN.

3 Bayesian Design Method for LLWNN

3.1 Bayesian Method

First of all, D, x, y and θ denote observed data $D = (x, y)$, training data, desired data and unknown parameter, respectively. The Bayesian method follows the following procedure [18].

(1) Modeling $p(\cdot|\theta)$
(2) Determine a prior distribution $p(\theta)$
(3) Observe data $p(D|\theta)$
(4) Calculate a posteriori distribution $p(\theta|D)$
(5) Estimate a predictive distribution $q(y|x, D)$

The posteriori distribution $p(\theta|D)$ is derived using the following Bayes' theorem.

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)}. \tag{4}$$

Then, if the complexity of model is known, the predictive distribution $q(y|x, D)$ is derived as

$$q(y|x, D) = \int p(y|x, \theta)p(\theta|D)d\theta, \tag{5}$$

and if the the complexity of the model, m, is unknown, it is derived as

$$q(y|x, D) = \sum_m \int p(y|x, \theta, m)p(\theta, m|D)d\theta. \tag{6}$$

The Bayesian method estimates the parameters of observed data to be approximated as the posteriori distribution of parameters [11]. Its characteristics are listed as follows.

- It can utilize a priori knowledge as the prior distribution.
- It shows high generalization ability even if training data is small.
- It can evaluate the reliability of the predictive value.
- It can automatically select models.

On the other hand, the Bayesian method has a serious problem on the integral calculation in posteriori distribution. It requires any approximation methods or any numerical calculation methods. So far, four method, (1) Laplace approximation method [18], (2) mean field method [19], (3) MCMC (Markov Chain Monte Carlo) method [20] and (4) variational Bayes method [21] have been proposed.

A method to be proposed in the present paper utilizes the above MCMC method. In the present paper, as both network design and parameter learning of LLWNN are conducted and then the size of model is unknown, the reversible jump MCMC method [20] is employed

3.2 Proposed Bayesian Design Method

In this section, the proposed Bayesian method for LLWNN is described. The proposed method follows a framework by C. C. Holmes et al. [16] and formulates a LLWNN. In the present paper, the LLWNN by applying the proposed method is called a BLLWNN (Bayesian Local Linear Wavelet Neural Network).

At first, data set $D = \{(\mathbf{x}_i, y_i) \,|\, i = 1 \sim n\}$ is defined by input data $\mathbf{x}_i \in R^N$ and output data $y_i \in R$. The relation between $\mathbf{x}_i$ and y_i is represented by

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \tag{7}$$

where ϵ_i referes to a noise term. After that, the input-output characteristic $f(\cdot)$ is modeled by (1). Then, a model M_k and model space M are written as

$$M_k = \{a_1, \mathbf{b}_1, \cdots, a_k, \mathbf{b}_k\}, \tag{8}$$

$$M = \{k, M_k, W\}, \tag{9}$$

respectively, where $W = (\mathbf{w}_1, \cdots, \mathbf{w}_k)$ means a weight matrix and a vector element $\mathbf{w}_k$ is written as $\mathbf{w}_k = (w_{k0}, w_{k1}, \cdots, w_{kN})'$ ($\mathbf{w}'$ refers to a transpose of $\mathbf{w}$). Furthermore, an unknown parameter θ_k of LLWNN is written as $\theta_k = \{a_k, \mathbf{b}_k, \mathbf{w}_k\}$.

The predictive distribution is derived by averaging predictive output $\hat{f}_M(\mathbf{x})$ with the posteriori distributions $p(W|M_k, D)$ and $p(M_k|D)$.

$$p(y|\mathbf{x}, D) = \sum_k \iint \hat{f}_M(\mathbf{x}) p(W|M_k, D) p(M_k|D) dM_k dW, \tag{10}$$

where $p(W|M_k, D)$ and $p(M_k|D)$ are written as

$$p(W|M_k, D) = \frac{p(D|W, M_k)p(W)}{p(D)}, \tag{11}$$

$$p(M_k|D) = \frac{p(D|M_k)p(M_k)}{p(D)}. \tag{12}$$

In (7), if the noise term ϵ_i follows a normal distribution with mean 0 and variation σ^2, $p(D|W, M_k)$ in (11) is derived as

$$\begin{aligned} p(D|W, M_k) &= \log \prod_{i=1}^{N} p(\epsilon_i) \\ &= \log \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - f(\mathbf{x}_i, M_k, W))^2}{2\sigma^2} \right) \\ &= -\frac{n}{2}\log(2\pi) - n\log\sigma - \frac{1}{2\sigma^2}\sum_{i=1}^{N} \{y_i - f(\mathbf{x}_i, M_k, W)\}^2 . \end{aligned} \tag{13}$$

In the present paper, prior distributions $p(W)$ in (11) and $p(M)$ in (12) are assumed by the following normal and gamma distributions, respectively.

$$p(W) = N(W|0, \lambda^{-1}I), \tag{14}$$

$$p(M) = Ga(DF|\alpha, \beta), \tag{15}$$

where DF means a degrees of freedom of model and is defined by

$$DF = tr\left(\Psi(\Psi'\Psi + \lambda I)^{-1}\Psi'\right), \tag{16}$$

where $tr(S)$ refers to a trace of matrix S. From (14) and (15),

$$\begin{aligned} p(M, W) &= Ga(DF|\alpha, \beta)N(W|0, \lambda^{-1}I) \\ &\propto DF^{\alpha-1}\exp(-\beta DF)\exp\left(-\frac{\lambda}{2}||W||^2\right). \end{aligned} \tag{17}$$

In the present paper, the integral calculation in (10) is solved using the reversible jump MCMC method, which is a type of MCMC that allows for dimensional changes in the probability distribution being simulated [20]. Therefore, (10) is approximated by

$$p(y|\mathbf{x}, D) \approx \frac{1}{n_s - n_0} \sum_{t=n_0}^{n_s} f_{\pi_t}(\mathbf{x}), \tag{18}$$

where π_t is a Markov chain sample followed by the probability distribution $p(\pi|D)$, n_0 is the number of samples in burn-in time and n_s is the total number of samples. In the present paper, a Metropolis-Hastings algorithm [11] is used for sampling.

4 Computer Simulation

This section describes computer simulation to evaluate the performance of the proposed method.

In the simulation, a function approximation problem was used. The following two-dimensional continuous function used by T. Wang et al. [7] was employed.

$$f(x_1, x_2) = \frac{\sin(\pi x_1)\cos(\pi x_2) + 1.0}{2.0}, \tag{19}$$

where the domains of input variables x_1 and x_2 were set to $x_1,\ x_2 \in [-1.0, 1.0]$. The training data set consists of 49 input points (x_1, x_2) which generated by equally spaced on a 7×7 grid in $[-1.0, 1.0] \times [-1.0, 1.0]$ and corresponding output $f(x_1, x_2)$. Then, testing data set consists of 400 input points which generated by equally spaced on a 20×20 grid in the same domain.

The basis function, i.e. ψ in (2) is defined by

$$\psi(x) = -x \exp\left(-\frac{x^2}{2}\right). \tag{20}$$

Then, function Ψ is written as tensor product of ψ for input x_i as follows.

$$\Psi(\mathbf{x}) = \prod_{i=1}^{N} \psi(x_i). \tag{21}$$

The performance was evaluated by three factors, i.e. the number of hidden units, the number of free parameters and the root-mean-square error (RMSE) defined by

$$\mathrm{RMSE} = \sqrt{\frac{1}{N_p} \sum_{j=1}^{N_p} \left\{ f(x_j) - \hat{f}(x_j) \right\}^2}, \tag{22}$$

where $f(x_j)$ and $\hat{f}(x_j)$ are desired and predictive values for input x_j, respectively and N_p is the number of training data.

Table 1. Parameter setting

Parameter	Value
σ	1.0
λ	0.01
α	0.1
β	0.1
n_s	10000
n_0	5000

Table 2. Simulation result

Method	# of hidden units	# of free parameters	RMSE
WNN	8	40	1.65×10^{-2}
LLWNN	4	28	1.58×10^{-2}
BLLWNN	4	28	1.56×10^{-2}

The parameter setting and the simulation results are shown in Table 1 and 2, respectively. In Table 2, the results of general WNN and LLWNN [7] are also provided to compare with the proposed method, i.e. BLLWNN. Both WNN and LLWNN empirically determine the number of hidden units. The BLLWNN, however, could automatically determine the numbers of hidden units and free parameters because of the Bayesian method.

As shown in Table 2, BLLWNN is much better than WNN. Furthermore, the BLLWNN shows almost the same performance with LLWNN. Therefore, BLLWNN is superior than LLWNN and WNN because it could automatically build network structure and determine its parameters.

5 Summary

In the present paper, the Bayesian method to determine both network structure and parameter of LLWNN has been proposed. Through computer simulation using function approximation problem, the effectiveness of the proposed method is confirmed. The evaluation for higher-dimensional functions and time-series prediction are future work.

References

1. Chui, C.K.: An Introduction to Wavelets. Academic Press, London (1992)
2. Poggio, T., Girosi, F.: Networks for approximation and learning. Proc. of the IEEE 78(9), 1481–1497 (1990)
3. Pati, Y.C., Krishnaprasad, P.S.: Discrete affine wavelet transformations for analysis and synthesis of feedforward neural networks. In: Advances in Neural Information Processing Systems, vol. 3, pp. 743–749. MIT Press, Cambridge (1991)

4. Zhang, Q., Benveniste, A.: Wavelet networks. IEEE Trans. on Neural Networks 3(6), 889–898 (1992)
5. Kobayashi, K., Torioka, T., Yoshida, N.: A Wavelet Neural Network with Network Optimizing Function. Systems and Computers in Japan 26(9), 61–71 (1995)
6. Ueda, N., Kobayashi, K., Torioka, T.: A Wavelet Neural Network with Evolutionally Generated Structures. Trans. on the Institute of Electronics, Information and Communication Engineers J80-D-II(2), 652–659 (1997) (in Japanese)
7. Wang, T., Sugai, Y.: A local linear adaptive wavelet neural network. Trans. on the Institute of Electrical Engineers of Japan 122-C(2), 277–284 (2002)
8. Wang, T., Sugai, Y.: The local linear adaptive wavelet neural network with hybrid ep/gradient algorithm and its application to nonlinear dynamic system identification. Trans. on the Institute of Electrical Engineers of Japan 122-C(7), 1194–1201 (2002)
9. Chen, Y., Dong, J., Yang, B., Zhang, Y.: A local linear wavelet neural network. In: Proc. of the 5th World Congress on Intelligent Control and Automation, pp. 1954–1957 (2004)
10. Chen, Y., Yang, B., Dong, J.: Time-series prediction using a local linear wavelet neural network. Neurocomputing 69, 449–465 (2006)
11. MacKay, D.: Information Theory, Inference, and Learning Algorithms. Cambridge University Press, Cambridge (2003)
12. Albrecht, S., Busch, J., Kloppenburg, M., Metze, F., Tavan, P.: Generalized radial basis function networks for classification and novelty detection: Self-organization of optimal Bayesian decision. Neural Networks 13, 755–764 (2000)
13. Andrieu, C., de Freitas, N., Doucet, A.: Robust full Bayesian learning for radial basis networks. Neural Computation 13, 2359–2407 (2001)
14. Ueda, N., Nakano, R., Ghahramani, Z., Hinton, G.E.: Split and merge EM algorithm for improving Gaussian mixture density estimates. In: Proc. of the 1998 IEEE Signal Processing Society Workshop, pp. 274–283 (1998)
15. Sato, M.: Online model selection based on the variational Bayes. Neural Computation 13, 1649–1681 (2001)
16. Holmes, C.C., Mallick, B.K.: Bayesian radial basis functions of variable dimension. Neural Computation 10, 1217–1233 (1998)
17. Sastry, K., Goldberg, D.E.: Probabilistic Model Building and Competent Genetic Programming, pp. 205–220. Kluwer, Dordrecht (2003)
18. MacKay, D.J.C.: Bayesian interpolation. Neural Computation 4, 415–447 (1992)
19. Peterson, C., Anderson, J.R.: A Mean Field Theory Learning Algorithm for Neural Networks. Complex Systems 1, 995–1019 (1987)
20. Green, P.J.: Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika 82, 711–732 (1995)
21. Attias, H.: Inferring parameter and structure of latent variable models by variational Bayes. In: Proc. of the 15th Conf. on Uncertainty in Artificial Intelligence, pp. 21–30 (1999)

Analysis on Equilibrium Point of Expectation Propagation Using Information Geometry

Hideyuki Matsui and Toshiyuki Tanaka

Graduate School of Informatics, Kyoto University
Kyoto, 606-8501 Japan
{matsuih@sys.,tt@}i.kyoto-u.ac.jp

Abstract. Expectation Propagation (EP) extends belief propagation by approximating messages with expectations of statistics, in which users can choose the statistics. In this paper, we discuss how a choice of statistics affects accuracy of EP's estimates. We approximate estimation error of EP via perturbation analysis based on information geometry. By comparing the approximated estimation error, we show that adding statistics does not necessarily improve the accuracy of EP. A numerical example confirms validity of our analytical results.

1 Introduction

Belief Propagation (BP) has recently been drawing attention as a computationally efficient approximation method for probabilistic inference problems. BP updates and passes marginal distributions as messages, which however causes difficulty in applying BP to inference problems with continuous variables. Expectation Propagation (EP) [1] extends BP by making messages to be expectations of statistics [1,2,3], which allows EP to treat continuous variables efficiently. It also gives users a chance of choosing statistics which define EP's messages. The freedom of choice of statistics raises another problem of EP: How does the choice affect accuracy of EP's estimates?

In this paper, we tackle this problem by using information geometry [4], which has been successfully applied to analysis of estimation errors of BP [5,6]. Viewing EP as an extension of BP, in Sect. 2 we reformulate EP by following the information-geometrical framework of BP [5,6]. In Sect. 3 we approximate EP's estimation error via perturbation analysis and argue its dependence on a choice of statistics. We supplement our analytical result with a numerical example in Sect. 4.

2 Information Geometrical View of EP

Let $\boldsymbol{x} = (x_1, \cdots, x_N)$ be a random vector and let $q(\boldsymbol{x})$ be its probability distribution, which we call a *target distribution.* We define $\{f_i(\boldsymbol{x})\}$ as a set of real-valued functions of $\boldsymbol{x}$. Assume that we are interested in expectation of $f_i(\boldsymbol{x})$ with respect to the target distribution, that is, $\eta_i(q) = \langle f_i(\boldsymbol{x})\rangle_q = \sum_{\boldsymbol{x}} f_i(\boldsymbol{x})q(\boldsymbol{x})$. This

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 155–162, 2009.

situation often appears in machine learning as well as several other Bayesian inference problems. An example of the situation is posterior mean estimation: $f_i(\boldsymbol{x}) = x_i$ for $i = 1, \cdots, N$ and $q(\boldsymbol{x})$ is a posterior distribution. If $\boldsymbol{x}$ is of high dimension and q is "intractable," it is difficult to obtain $\eta_i(q)$. In this paper "a distribution $p(\boldsymbol{x})$ is intractable" means that one has to sum up $f_i(\boldsymbol{x})p(\boldsymbol{x})$ for all configurations of $\boldsymbol{x}$ in order to evaluate $\eta_i(p)$. Conversely, "$p(\boldsymbol{x})$ is tractable" means that one can evaluate $\eta_i(p)$ analytically or one does not have to consider all the configuration of $\boldsymbol{x}$ in order to evaluate it due to its independence structure. If the target distribution is intractable, one requires some kind of approximation method. EP is one of such methods.

We assume that the target distribution is decomposed as

$$q(\boldsymbol{x}) \propto \exp\left[c_1(\boldsymbol{x}) + c_2(\boldsymbol{x}) + \cdots + c_L(\boldsymbol{x})\right], \tag{1}$$

where $\{c_r(\boldsymbol{x})\}$ is a set of real-valued functions. We consider an exponential family

$$S = \{p(\boldsymbol{x};\, \boldsymbol{\theta},\, \boldsymbol{v}) = \exp[\theta^i f_i(\boldsymbol{x}) + v^r c_r(\boldsymbol{x}) - \psi(\boldsymbol{\theta},\, \boldsymbol{v})]\}, \tag{2}$$

where $(\boldsymbol{\theta},\, \boldsymbol{v})$ is a canonical coordinate system of S, and where $\psi(\boldsymbol{\theta},\, \boldsymbol{v})$ is for normalization. Here and hereafter, we use Einstein's convention: If an index appears as a superscript and a subscript within the same term, we take a summation with respect to the index. The target distribution has the coordinate $(\boldsymbol{\theta},\, \boldsymbol{v}) = (\mathbf{0},\, \mathbf{1})$, where $\mathbf{1}$ denotes an all-1 vector. We call the statistics $\{f_i(\boldsymbol{x})\}$ which define S the *basis statistics*. Given $q(\boldsymbol{x})$, an EP user can choose basis statistics arbitrarily, as long as they include statistics whose expectations are of interest.

We give a reformulation of EP, which forms a basis for our information geometrical argument. First, we define subfamilies S_0 and $\{S_r\}$ of S as

$$S_0 = \{p_0(\boldsymbol{x};\, \boldsymbol{\theta}) = \exp[\theta^i f_i(\boldsymbol{x}) - \psi_0(\boldsymbol{\theta})]\}, \tag{3}$$

$$S_r = \{p_r(\boldsymbol{x};\, \boldsymbol{\zeta}_r) = \exp[\zeta_r^i f_i(\boldsymbol{x}) + c_r(\boldsymbol{x}) - \psi_r(\boldsymbol{\zeta}_r)]\}, \quad r = 1, \ldots, L, \tag{4}$$

where $\boldsymbol{\zeta}_r$ is a canonical coordinate system of S_r. For EP to be computationally feasible, we assume that all distributions in $\{S_r\}$ are tractable and that evaluation of $\boldsymbol{\theta}(p_0)$ from $\boldsymbol{\eta}(p_0) = (\eta_1(p_0), \ldots)$ is easy, where $\boldsymbol{\theta}(p_0)$ is the value of the parameter $\boldsymbol{\theta}$ of a distribution $p_0 \in S_0$. The reformulation of EP shown below closely follows [5,6]:

1. Initialize $\boldsymbol{\theta}$ and $\{\boldsymbol{\xi}_r\}$. Let $\boldsymbol{\zeta}_r = \boldsymbol{\theta} - \boldsymbol{\xi}_r$ for all indexes r.
2. For one or several numbers of r, evaluate $\boldsymbol{\eta}(p_r)$ for the distributions $p_r(\boldsymbol{x};\, \boldsymbol{\zeta}_r)$.
3. Derive distributions $p_{0r} \in S_0$ which satisfy $\boldsymbol{\eta}(p_{0r}) = \boldsymbol{\eta}(p_r)$.
4. Evaluate $\boldsymbol{\theta}(p_{0r})$ from $\boldsymbol{\eta}(p_{0r})$. Let $\boldsymbol{\xi}_r := \boldsymbol{\theta}(p_{0r}) - \boldsymbol{\zeta}_r$.
5. Let $\boldsymbol{\theta} := \sum_r \boldsymbol{\xi}_r$ and $\boldsymbol{\zeta}_r := \boldsymbol{\theta} - \boldsymbol{\xi}_r$.
6. Repeat 2–5 until convergence.
7. Output $\boldsymbol{\eta}^* := \boldsymbol{\eta}(p_0)|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}$ as an estimate of $\boldsymbol{\eta}(q)$.

Here and hereafter, starred variables denote their equilibrium values.

In this paper we assume that EP converges, although the convergence is not guaranteed in general, and study properties of its equilibrium. Usually, EP's estimate $\boldsymbol{\eta}^*$ is not equal to $\boldsymbol{\eta}(q)$, and estimation error is affected by choices of the basis statistics. Intuitively, one would expect that the more statistics one uses as the basis statistics, the more the accuracy is. A similar problem in the context of Generalized Belief Propagation was discussed by Welling [7], which asks whether or not adding a new region always improves accuracy. In our context, he argued that the intuition is wrong if the target distribution is distant from S_0. He also argued that the intuition should be true if the target distribution is close to S_0, but the validity of his hypothesis remains a matter of research.

We tackle this problem by comparing EP's estimation errors with two different models studied in Sects. 3.2 and 3.3. The model studied in Sect. 3.2 is included in the model we consider in Sect. 3.3, so we call the former the small model and the latter the large model.

3 Accuracy of Estimates

3.1 Properties of Equilibrium

As observed in our reformulation in Sect. 2, EP corresponds to BP with a particular choice of basis statistics. It then follows that the equilibrium of EP satisfies the same conditions as those of BP, called the e- and the m-conditions [5,6],

$$e\textbf{-condition}: \ \boldsymbol{\theta}^* = \sum_r \boldsymbol{\xi}_r^* = -\sum_r (\boldsymbol{\zeta}_r^* - \boldsymbol{\theta}^*), \tag{5}$$

$$m\textbf{-condition}: \ \boldsymbol{\eta}^* \equiv \boldsymbol{\eta}(p_0)|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*} = \boldsymbol{\eta}(p_r)|_{\boldsymbol{\zeta}_r=\boldsymbol{\zeta}_r^*}. \tag{6}$$

The e-condition relates $p_0(\boldsymbol{x};\, \boldsymbol{\theta^*})$, $\{p_r(\boldsymbol{x};\, \boldsymbol{\zeta}_r^*)\}$ and $q(\boldsymbol{x})$: Let

$$E^* = \Big\{ p_E(\boldsymbol{x};\, t_0,\, \boldsymbol{t}) \,|\, \ln p_E(\boldsymbol{x};\, t_0,\, \boldsymbol{t}) = t_0 \ln p_0(\boldsymbol{x};\, \boldsymbol{\theta^*}) + \sum_{r=1}^{L} t_r \ln p_r(\boldsymbol{x};\, \boldsymbol{\zeta}_r^*) + C(t_0,\, \boldsymbol{t}),\, t_r \in \Re,\, \sum_{r=0}^{L} t_r = 1 \Big\}, \tag{7}$$

where $\boldsymbol{t} = (t_1, \cdots, t_L)^T$, and where $C(t_0,\, \boldsymbol{t})$ is a normalization term. E^* is a log-linear submodel of S connecting $p_0(\boldsymbol{x};\, \boldsymbol{\theta^*}) = p_E(\boldsymbol{x};\, 1,\, \boldsymbol{0})$ and $\{p_r(\boldsymbol{x};\, \boldsymbol{\zeta}_r^*) = p_E(\boldsymbol{x};\, 0,\, \boldsymbol{e}_r)\}$, where $\boldsymbol{e}_r$ is a unit vector whose rth component is 1. The e-condition dictates that E^* contains $q(\boldsymbol{x}) = p_E(\boldsymbol{x};\, -(L-1),\, \boldsymbol{1})$ as well.

3.2 Small Model

We define the small model S by choosing $\{f_i(\boldsymbol{x})\}$ as the basis statistics, as

$$S = \{p(\boldsymbol{x};\, \boldsymbol{\theta},\, \boldsymbol{v}) = \exp[\theta^i f_i(\boldsymbol{x}) + v^r c_r(\boldsymbol{x}) - \psi(\boldsymbol{\theta},\, \boldsymbol{v})]\}. \tag{8}$$

We evaluate the estimation error $\boldsymbol{\eta}(q) - \boldsymbol{\eta}^*$ via second-order perturbation analysis [5]. First, we introduce a foliation $\{S(\boldsymbol{v})\}$ of S with

$$S(\boldsymbol{v}) = \{p(\boldsymbol{x};\, \boldsymbol{\zeta},\, \boldsymbol{v}) = \exp[\zeta^i f_i(\boldsymbol{x}) + v^r c_r(\boldsymbol{x}) - \psi(\boldsymbol{\zeta},\, \boldsymbol{v})]\}. \tag{9}$$

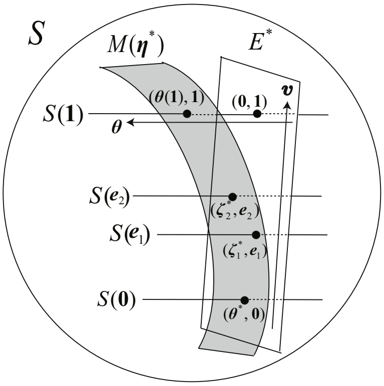

S
$M(\boldsymbol{\eta}^*)$
E^*
$S(\mathbf{1})$
$\boldsymbol{\theta}$
$(\boldsymbol{\theta}(\mathbf{1}),\mathbf{1})$
$(\mathbf{0},\mathbf{1})$
$\boldsymbol{v}$
$S(\boldsymbol{e}_2)$
$(\boldsymbol{\zeta}^*_2,\boldsymbol{e}_2)$
$S(\boldsymbol{e}_1)$
$(\boldsymbol{\zeta}^*_1,\boldsymbol{e}_1)$
$S(\mathbf{0})$
$(\boldsymbol{\theta}^*,\mathbf{0})$

Fig. 1. Relation between $(\boldsymbol{\theta}(\mathbf{1}),\ \mathbf{1})$ and $(\mathbf{0},\ \mathbf{1})$. Estimation error is $\eta_i(\mathbf{0},\ \mathbf{1})-\eta_i(\boldsymbol{\theta}(\mathbf{1}),\ \mathbf{1})$.

Note that $q(\boldsymbol{x}) \in S(\mathbf{1})$, $S_0 = S(\mathbf{0})$ and $S_r = S(\boldsymbol{e}_r)$ hold. We introduce another foliation $\{M(\boldsymbol{\eta})\}$ of S with $M(\boldsymbol{\eta}) = \{p \in S | \boldsymbol{\eta}(p) = \boldsymbol{\eta}\}$. From the m-condition, $M(\boldsymbol{\eta}^*)$ includes $\boldsymbol{\eta}(p_0)|_{\boldsymbol{\theta}=\boldsymbol{\theta}^*}$ and $\boldsymbol{\eta}(p_r)|_{\boldsymbol{\zeta}_r=\boldsymbol{\zeta}_r^*}$.

In the following, in order to avoid possible confusion due to our use of many indexes, we introduce a rule of using the indexes i, j and k for specifying components of $\boldsymbol{\theta}$, $\boldsymbol{\eta}$ and $\{\boldsymbol{\zeta}_r\}$, and r and s for specifying components of $\boldsymbol{v}$.

The key to evaluating the error is to study the intersection of $S(\mathbf{1})$ and $M(\boldsymbol{\eta}^*)$. On the submodel $M(\boldsymbol{\eta}^*)$ of S, the parameter $\boldsymbol{\theta}$ is an implicit function of $\boldsymbol{v}$, which is denoted by $\boldsymbol{\theta}(\boldsymbol{v})$. The intersection of $S(\mathbf{1})$ and $M(\boldsymbol{\eta}^*)$ has the coordinate $(\boldsymbol{\theta}(\mathbf{1}),\ \mathbf{1})$. We let $\eta_i(\boldsymbol{\theta},\ \boldsymbol{v})$ denote the expectation of $f_i(\boldsymbol{x})$ with respect to $p(\boldsymbol{x};\ \boldsymbol{\theta}, \boldsymbol{v})$, so that $\eta_i^* = \eta_i(\boldsymbol{\theta}(\mathbf{1}),\ \mathbf{1})$ and $\eta_i(q) = \eta_i(\mathbf{0},\ \mathbf{1})$ hold. The first-order Taylor expansion along $S(\mathbf{1})$ at $(\boldsymbol{\theta}(\mathbf{1}),\ \mathbf{1})$ yields the approximation of the error

$$\eta_i(q) - \eta_i^* \approx -g_{ij}^* \theta^j(\mathbf{1}), \tag{10}$$

where $g_{ij} = \partial\eta_i/\partial\theta^j$. Figure 1 shows the relation between $(\boldsymbol{\theta}(\mathbf{1}),\ \mathbf{1})$ and $(\mathbf{0},\ \mathbf{1})$.

To evaluate $\boldsymbol{\theta}(\mathbf{1})$, we approximate $\boldsymbol{\theta}(\boldsymbol{v})$ by the second-order Taylor expansion around $\boldsymbol{v} = \mathbf{0}$ and let $\boldsymbol{v} = \mathbf{1}$, yielding

$$\theta^i(\mathbf{1}) \approx \theta^{i*} + \sum_r A_r^{i*} + \frac{1}{2}\sum_{r,\ s} A_{rs}^{i*}. \tag{11}$$

where $A_r^{i*} = \partial\theta^i/\partial v^r|_{\boldsymbol{v}=\mathbf{0}}$ and $A_{rs}^{i*} = \partial^2\theta^i/\partial v^r \partial v^s|_{\boldsymbol{v}=\mathbf{0}}$. Similarly, one has

$$\zeta_r^{i*} = \theta^i(\boldsymbol{e}_r) \approx \theta^{i*} + A_r^{i*} + \frac{1}{2}A_{rr}^{i*}. \tag{12}$$

Substituting (12) into the right-hand side (RHS) of the e-condition (5), we obtain

$$\theta^{i*} = -\sum_r(\zeta_r^{i*} - \theta^{i*}) \approx -\sum_r \left(A_r^{i*} + \frac{1}{2}A_{rr}^{i*}\right). \tag{13}$$

Substituting (13) into (11), one obtains the approximation

$$\theta^i(\mathbf{1}) \approx \frac{1}{2} \sum_{\substack{r,\, s \\ r \neq s}} A^{i*}_{rs}. \tag{14}$$

Since derivatives of $\boldsymbol{\eta}$ with respect to $\boldsymbol{v}$ along $M(\boldsymbol{\eta}^*)$ should vanish identically, considering the derivatives around $(\boldsymbol{\theta}^*,\, \mathbf{0})$ leads to

$$g^*_{ij} A^{j*}_r + g^*_{ir} = 0, \tag{15}$$

$$g^*_{ij} A^{j*}_{rs} = -B_r B_s \eta_i, \tag{16}$$

where B_r is an operator which evaluates a derivative $\partial_r + A^{j*}_r \partial_j$ at $(\boldsymbol{\theta}, \boldsymbol{v}) = (\boldsymbol{\theta}^*, \mathbf{0})$, and where $g_{ir} = \partial \eta_i / \partial v_r$. Finally, from (10), (14), and (16), the approximation of estimation error is evaluated as

$$\eta_i(q) - \eta^*_i \approx \frac{1}{2} \sum_{\substack{r,\, s \\ r \neq s}} B_r B_s \eta_i. \tag{17}$$

Since our argument is based on Taylor expansion, the closer the target distribution is to S_0, the more accurate our approximation is expected to be.

3.3 Large Model

We define the large model, denoted by $\bar{S}$, as

$$\bar{S} = \{p(\boldsymbol{x};\, \boldsymbol{\theta},\, \boldsymbol{\delta},\, \boldsymbol{v}) = \exp[\theta^i f_i(\boldsymbol{x}) + \delta^\lambda g_\lambda(\boldsymbol{x}) + v^r c_r(\boldsymbol{x}) - \bar{\psi}(\boldsymbol{\theta},\, \boldsymbol{\delta},\, \boldsymbol{v})]\} \tag{18}$$

where $\{g_\lambda(\boldsymbol{x})\}$ is a set of real-valued functions of $\boldsymbol{x}$, and where $(\boldsymbol{\theta},\, \boldsymbol{\delta},\, \boldsymbol{v})$ is a canonical coordinate system of $\bar{S}$. The basis statistics of the model $\bar{S}$ are $\{f_i(\boldsymbol{x})\} \cup \{g_\lambda(\boldsymbol{x})\}$. The small model S is a submodel of $\bar{S}$ with $\boldsymbol{\delta} = \mathbf{0}$. Here and hereafter, we use the indexes λ and κ for components of $\boldsymbol{\delta}$ and $\boldsymbol{\mu}$, in addition to the rule introduced in Sect. 3.2. We let $\mu_\lambda(p)$ denote expectation of $g_\lambda(\boldsymbol{x})$ with respect to $p(\boldsymbol{x})$, and let $\boldsymbol{\mu}(p) = (\mu_1(p), \ldots)$.

We follow the same procedure as in Sect. 3.2 to evaluate estimation error of EP in $\bar{S}$. We introduce the dual foliations $\{\bar{S}(\boldsymbol{v})\}$ and $\{\bar{M}(\boldsymbol{\eta},\, \boldsymbol{\mu})\}$ of $\bar{S}$, with

$$\bar{S}(\boldsymbol{v}) = \{p(\boldsymbol{x};\, \boldsymbol{\zeta},\, \boldsymbol{\tau},\, \boldsymbol{v}) = \exp[\zeta^i f_i(\boldsymbol{x}) + \tau^\lambda g_\lambda(\boldsymbol{x}) + v^r c_r(\boldsymbol{x}) - \bar{\psi}(\boldsymbol{\zeta},\, \boldsymbol{\tau},\, \boldsymbol{v})]\} \tag{19}$$

and $\bar{M}(\boldsymbol{\eta},\, \boldsymbol{\mu}) = \{p \in \bar{S} | (\boldsymbol{\eta}(p),\, \boldsymbol{\mu}(p)) = (\boldsymbol{\eta},\, \boldsymbol{\mu})\}$, and define $\bar{S}_0 := \bar{S}(\mathbf{0})$ and $\bar{S}_r := \bar{S}(\boldsymbol{e}_r)$. We assume that all distributions in $\bar{S}_r$ are tractable and that one can evaluate $(\boldsymbol{\theta}(p_0),\, \boldsymbol{\delta}(p_0))$ from $(\boldsymbol{\eta}(p_0),\, \boldsymbol{\mu}(p_0))$ easily for all $p_0 \in \bar{S}_0$.

The result in Sect. 3.2 is applicable to $\bar{S}$ as well, yielding

$$\eta_i(q) - \bar{\eta}^*_i \approx -\bar{g}^*_{ij} \theta^j(\mathbf{1}) - \bar{g}^*_{i\kappa} \delta^\kappa(\mathbf{1}) \approx \frac{1}{2} \sum_{\substack{r,\, s \\ r \neq s}} \bar{B}_r \bar{B}_s \eta_i, \tag{20}$$

where $\bar{B}_r$ is an operator which evaluates a derivative $\partial_r + \bar{A}^{j*}_r \partial_j + \bar{A}^{\kappa *}_r \partial_\kappa$ at $(\boldsymbol{\theta}^*, \boldsymbol{\delta}^*, \mathbf{0})$. The derivatives $\bar{A}^{i*}_r = \partial \theta^i / \partial v^r |_{\boldsymbol{v}=\mathbf{0}}$ and $\bar{A}^{\lambda *}_r = \partial \delta^\lambda / \partial v^r |_{\boldsymbol{v}=\mathbf{0}}$ satisfy

$$\begin{pmatrix} \bar{g}^*_{ij} & \bar{g}^*_{i\kappa} \\ \bar{g}^*_{\lambda j} & \bar{g}^*_{\lambda\kappa} \end{pmatrix} \begin{pmatrix} \bar{A}^{j*}_r \\ \bar{A}^{\kappa *}_r \end{pmatrix} + \begin{pmatrix} \bar{g}^*_{ir} \\ \bar{g}^*_{\lambda r} \end{pmatrix} = \begin{pmatrix} O \\ O \end{pmatrix}. \tag{21}$$

3.4 Comparison of Estimation Errors

We discuss effects of using extra statistics in the basis statistics by comparing estimation errors derived in Sects. 3.2 and 3.3. For simplicity, we assume that $\{g_\lambda(\boldsymbol{x})\}$ are orthogonal to $\{f_i(\boldsymbol{x})\}$ at $(\boldsymbol{\theta}, \boldsymbol{\delta}, \boldsymbol{v}) = (\boldsymbol{\theta}^*, \boldsymbol{\delta}^*, \mathbf{0})$, or equivalently, that $\bar{g}_{i\lambda}^* = 0$ holds. One has $\bar{A}_r^{i*} \approx A_r^{i*}$ in the leading order, and thus $\bar{B}_r \approx B_r + \bar{A}_r^{\kappa*}\partial_\kappa$. One can use it to relate the terms in (20) with those in (17), as

$$\bar{B}_r \bar{B}_s \eta_i \approx B_r B_s \eta_i + (\bar{A}_s^{\lambda*} B_r + \bar{A}_r^{\lambda*} B_s)\partial_\lambda \eta_i + \bar{A}_r^{\lambda*} \bar{A}_s^{\kappa*} \partial_\lambda \partial_\kappa \eta_i, \tag{22}$$

or in a more concrete form,

$$\begin{aligned}\bar{B}_r \bar{B}_s \eta_i \approx\ & T_{rsi} + \bar{A}_r^{j*} T_{sji} + \bar{A}_s^{j*} T_{rji} + \bar{A}_r^{j*} \bar{A}_s^{k*} T_{jki} \\ & + \bar{A}_s^{\lambda*}(T_{r\lambda i} + \bar{A}_r^{j*} T_{j\lambda i}) + \bar{A}_r^{\lambda*}(T_{s\lambda i} + \bar{A}_s^{j*} T_{j\lambda i}) + \bar{A}_r^{\lambda*} \bar{A}_s^{\kappa*} T_{\lambda\kappa i},\end{aligned} \tag{23}$$

where $T_{rsi} = \partial\eta_i/\partial v_r \partial v_s$, $T_{sji} = \partial\eta_i/\partial v_s \partial\theta_j$, $T_{jki} = \partial\eta_i/\partial\theta_j\partial\theta_k$, $T_{r\lambda i} = \partial\eta_i/\partial v_r \partial\delta_\lambda$, $T_{j\lambda i} = \partial\eta_i/\partial\theta_j\partial\delta_\lambda$ and $T_{\lambda\kappa i} = \partial\eta_i/\partial\delta_\lambda\partial\delta_\kappa$, all evaluated at $(\boldsymbol{\theta}^*, \boldsymbol{\delta}^*, \mathbf{0})$. The terms in the second line of the RHS of (23) represent effects of adding the statistics $\{g_\lambda(\boldsymbol{x})\}$. Since these terms can be positive or negative depending on choices of added statistics, we have shown that adding more statistics to the basis statistics does not necessarily improve the accuracy of EP. Note that our argument is valid when the target distribution is close to S_0.

4 Numerical Example

4.1 Formulation

In this section we supplement the result in the previous section with an example in which adding statistics to basis statistics results in loss of accuracy of EP. The target distribution in the example is a 3-variable Boltzmann machine

$$q(\boldsymbol{x};\, w) \propto \exp\left[w \sum_{i<j} x_i x_j + \sum_{i=1}^{3} x_i \right], \tag{24}$$

where $\boldsymbol{x} = (x_1, x_2, x_3) \in \{-1, 1\}^3$. We assume the decomposition $q(\boldsymbol{x}) \propto \exp\left[c(\boldsymbol{x}) + w\sum_{r=1}^3 c_r(\boldsymbol{x})\right]$ with

$$\begin{aligned} c(\boldsymbol{x}) &= x_1 + x_2 + x_3, & c_1(\boldsymbol{x}) &= 2x_1x_2 - x_1x_3, \\ c_2(\boldsymbol{x}) &= 2x_1x_3 - x_2x_3, & c_3(\boldsymbol{x}) &= 2x_2x_3 - x_1x_2, \end{aligned} \tag{25}$$

and consider the problem of estimating $\eta_1(q) = \langle x_1 \rangle_q$.

The small model S is defined as

$$S = \{p(\boldsymbol{x};\, \theta, \boldsymbol{v}) = \exp\left[c(\boldsymbol{x}) + \theta x_1 + v^r c_r(\boldsymbol{x}) - \psi(\theta, \boldsymbol{v})\right]\}, \tag{26}$$

where $(\theta, \boldsymbol{v})$ is a canonical coordinate system of S. The target distribution has the coordinate $(\theta, \boldsymbol{v}) = (0, w\mathbf{1})$. Application of our perturbation analysis to EP in S is straightforward, by absorbing $e^{c(\boldsymbol{x})}$ into the dominating measure. S is the simplest model to estimate $\eta_1(q)$, since its basis statistic is x_1 only. We define the submodels of S as

$$S_0 = \{p_0(\boldsymbol{x};\,\theta) = \exp\left[c(\boldsymbol{x}) + \theta x_1 - \psi_0(\theta)\right]\}, \tag{27}$$

$$S_r = \{p_r(\boldsymbol{x};\,\zeta_r) = \exp\left[c(\boldsymbol{x}) + \zeta_r x_1 + w c_r(\boldsymbol{x}) - \psi_r(\zeta_r)\right]\}, \tag{28}$$

where ζ_r is a canonical coordinate of S_r.

The large model $\bar{S}$ is defined by adding x_2 and x_3 to the basis statistics, as

$$\bar{S} = \{p(\boldsymbol{x};\,\boldsymbol{\theta},\,\boldsymbol{v}) = \exp\left[c(\boldsymbol{x}) + \theta^i x_i + v^r c_r(\boldsymbol{x}) - \bar{\psi}(\boldsymbol{\theta},\,\boldsymbol{v})\right]\}, \tag{29}$$

where $(\boldsymbol{\theta},\,\boldsymbol{v})$ is a canonical coordinate system of $\bar{S}$. The target distribution has the coordinate $(\boldsymbol{\theta},\,\boldsymbol{v}) = (\boldsymbol{0},\,w\boldsymbol{1})$. Note that EP in $\bar{S}$ corresponds to the conventional BP. We define the submodels of $\bar{S}$ as

$$\bar{S}_0 = \left\{p_0(\boldsymbol{x};\,\boldsymbol{\theta}) = \exp\left[c(\boldsymbol{x}) + \theta^i x_i - \bar{\psi}_0(\boldsymbol{\theta})\right]\right\}, \tag{30}$$

$$\bar{S}_r = \{p_r(\boldsymbol{x};\,\boldsymbol{\zeta}_r) = \exp\left[c(\boldsymbol{x}) + \zeta_r^i x_i + w c_r(\boldsymbol{x}) - \bar{\psi}_r(\boldsymbol{\zeta}_r)\right]\}, \tag{31}$$

where $\boldsymbol{\zeta}_r = (\zeta_r^1,\,\zeta_r^2,\,\zeta_r^3)$ is a canonical coordinate system of $\bar{S}_r$. In these settings, the parameter w determines how close the target distribution is to S_0 or $\bar{S}_0$.

4.2 Results

Figure 2 depicts how the EP's estimates vary as the parameter w changes. It shows that the EP's estimate in the large model is less accurate than that in the small model. Figure 3 shows the estimation errors and their second-order approximations derived in Sect. 3. It shows that the estimation errors are well

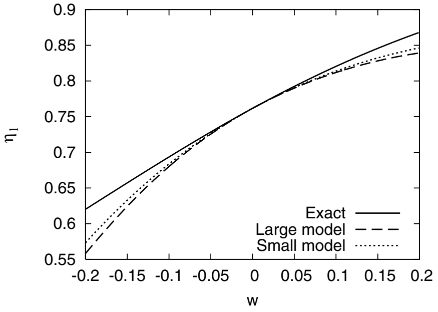


Fig. 2. Dependence of η_1 on the parameter w of the target distribution (24). Solid line shows the exact value of η_1. Dashed and dotted lines show the EP's estimates in the large and small models, respectively.

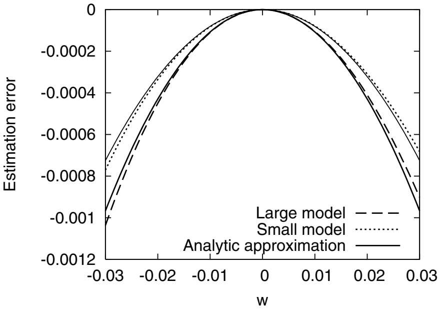


Fig. 3. EP's estimation errors of η_1. Dashed and dotted lines show the EP's estimation errors in the large and small models, respectively. Lower and upper solid lines show the second-order approximation in the large and small models, respectively.

predicted by our analysis around $w = 0$, confirming validity of our analysis. These observations support the result of second-order perturbation analysis that choosing more statistics as basis statistics does not always improve the accuracy of EP even when the target distribution is close to S_0. We would like to mention that we have also found examples in which the large model is less accurate basically because the target distribution is distant from S_0. But this example demonstrates that the large model can be less accurate even when w, which measures how distant the target distribution is from S_0, is small.

5 Conclusion

We have shown, by an information-geometry-based perturbation analysis, as well as by a numerical example, that adding more statistics to basis statistics does not necessarily improve the accuracy of EP, even when the target distribution is close to S_0, which implies that EP does not have a simple trade-off between model complexity and estimation accuracy.

It should be noted that this conclusion is not related with the *bias-variance trade-off*: The latter is regarding errors of estimating parameter values from finite-sized training sets, dictating that a larger model yields a smaller bias but suffers from a larger variance. On the other hand, we have discussed approximate estimation of expectations on the basis of a probability model with prespecified parameter values. Therefore, the framework of these problems are quite different.

Finally, we want to mention that our formula for estimation errors allows us to perform correction to EP's estimates. The validity and efficiency of this approach, however, need further investigation.

References

1. Minka, T.: A Family of Algorithms for Approximate Bayesian Inference. Ph.D. thesis, Massachusetts Institute of Technology (2001)
2. Heskes, T., Opper, M., Wiegerinck, W., Winther, O., Zoeter, O.: Approximate inference techniques with expectation constraints. J. Stat. Mech.: Theory Exp., P11015 (2005)
3. Wainwright, M.J.: Stochastic Processes on Graphs with Cycles. Ph.D. thesis, Massachusetts Institute of Technology (2002)
4. Amari, S., Nagaoka, H.: Methods of Information Geometry. Translations of Mathematical Monographs, vol. 191. American Mathematical Society (2001)
5. Ikeda, S., Tanaka, T., Amari, S.: Information geometry of turbo and low-density parity-check codes. IEEE Trans. Info. Theory 50, 1097–1114 (2004)
6. Ikeda, S., Tanaka, T., Amari, S.: Stochastic reasoning, free energy, and information geometry. Neural Comput. 16, 1779–1810 (2004)
7. Welling, M.: On the choice of regions for generalized belief propagation. In: Proc. 20th Conf. Uncertainty in AI, pp. 585–592 (2004)

Partially Enhanced Competitive Learning

Ryotaro Kamimura

IT Education Center, Tokai University
1117 Kitakaname Hiratsuka Kanagawa 259-1292, Japan
ryo@cc.u-tokai.ac.jp

Abstract. In this paper, we propose a new method to extract explicit features for competitive learning as well as self-organizing maps. The method aims to enhance final internal representations by conventional methods. We first train networks by conventional methods and compute enhanced information by focusing upon some specific input units or variables. Because we focus upon some specific inputs and activate competitive units, this enhancement is called *partial enhancement*. Then, networks are retrained to imitate the states obtained by partial enhancement. Final representations obtained by this retraining generate representations influenced by these specific variables. We applied the method to the famous Iris problem and the air pollution problem. In both problems, partial enhancement methods could produce clearer feature maps, superior to those obtained by self-organizing maps.

1 Introduction

In this paper, we propose a new method to produce explicit features or explicit self-organizing maps. Competitive learning and, especially, self-organizing maps by Kohonen have been used for clustering, feature detection and particularly for visualizing complex data. To obtain clearer maps, many visualization techniques have been developed [1], [2], [3], [4]. However, even with these sophisticated visualization techniques, it may be difficult to extract important features from the ambiguous representations generated by the conventional methods.

In this context, we introduce a new method to retrain networks to obtain clearer final internal representations. In this model, we have three steps. First, connection weights are obtained, for example, by using conventional SOM; then we compute enhanced information for all variables to detect which variables play important roles in feature detection. Finally, we try to imitate networks in which competitive units are enhanced by using specific variables with high enhanced information. Because final representations can be modified by influential variables in input patterns, conventional visualization techniques can more easily be applied to the representations to extract important features.

2 Theory and Computational Methods

2.1 Partial Enhancement

Figure 1 shows a concept of partial enhancement. We call enhancement ”partial enhancement” because some variables are partially used to enhance competitive

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 163–170, 2009.

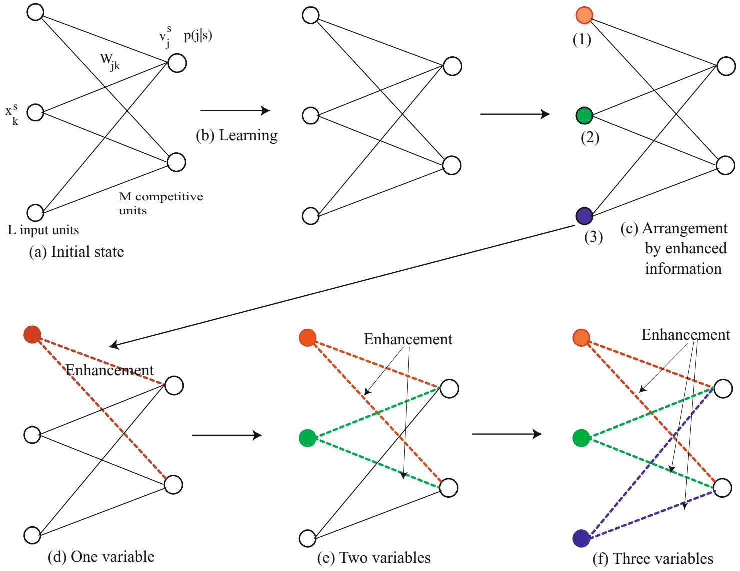


Fig. 1. A concept of partially enhanced competitive learning

units. First, a network is trained by conventional competitive learning or SOM, as shown in Figure 1(a). Then, by computing enhanced information for each variable, the variables are arranged according to the magnitude of enhanced information, as shown in Figure 1(c). Enhanced information is obtained by computing cross entropy between probabilities of firing of competitive units and probabilities obtained by partial enhancement. In Figure 1(c), enhanced information for the first variable is the highest. At the next stage, Figure 1(d), a network is retrained to imitate enhanced probabilities with one variable with maximum enhanced information. The number of variables used for retraining is increased gradually according to the magnitude of the enhanced information of the variables, as shown in Figure 1(e) and (f).

2.2 Enhanced Information for Input Units

Figure 1(a) shows a network architecture in which x_k^s denotes the kth element of the sth input pattern, and w_{jk} represents connection weights from the kth input unit to the jth competitive units. We now define enhanced information for input units. Distance between input units and connection weights when the kth input unit is used for enhancement is given by

$$v_{jt}^s = \exp\left(- \frac{\sum_{k=1}^{L} (x_k^s - w_{jk})^2}{2\sigma_{kt}^2} \right), \tag{1}$$

where

$$\sigma_{kt} = \begin{cases} 1/\alpha, & \text{if } k = t \ ; \\ \alpha, & \text{otherwise}, \end{cases}$$

where α is an enhancement parameter. For all experiments discussed in this paper, the enhanced parameter α is five. We can normalize these activations, and we have enhanced probabities

$$p^t(j \mid s) = \frac{v_{jt}^s}{\sum_{m=1}^{M} v_{mt}^s}. \tag{2}$$

By using these probabilities, we have enhanced information for the tth competitive unit

$$EI^t = \sum_{s=1}^{S}\sum_{j=1}^{M} p(s)p(j \mid s) \log \frac{p(j \mid s)}{p^t(j \mid s)}. \tag{3}$$

We examined the properties of this enhanced information and found that, as enhanced information is increased, the corresponding variables play more important roles in feature detection.

2.3 Enhanced Learning

Learning consists of imitating enhanced probabilities as much as possible. Now, $p(j \mid s)$ denotes the probability of firing of the jth competitive unit, and then we have cross entropy defined by

$$I = \sum_{s=1}^{S} p(s) \sum_{j=1}^{M} p(j \mid s) \log \frac{p(j \mid s)}{p^t(j \mid s)}. \tag{4}$$

It is possible to differentiate this cross entropy to obtain exact update rules [5], [6], [7]. However, the update rules need heavy computation for conditional probabilities. To simplify computational procedures, we follow statistical mechanics and introduce free energy [8], [9]. As is the free energy in statistical mechanics, free energy can be defined by

$$F = -2\sigma^2 \sum_{s=1}^{S} p(s) \log \sum_{j=1}^{M} p^t(j|s) \exp\left(-\frac{d_j^s}{2\sigma^2}\right). \tag{5}$$

For an optimal state, this equation is transformed into

$$F = \sum_{s=1}^{S} p(s) \sum_{j=1}^{M} p(j \mid s) \sum_{k=1}^{L} (x_k^s - w_{jk})^2 + 2\sigma^2 \sum_{s=1}^{S} p(s) \sum_{j=1}^{M} p(j \mid s) \log \frac{p(j \mid s)}{\tilde{p}(j \mid s)}. \tag{6}$$

This equation shows that, to minimize the free energy, we must minimize the cross entropy as well as the error function. Now, it is easy to differentiate the free energy, and we have

$$\Delta w_{jk} = \beta \sum_{s=1}^{S} p(s)p(j \mid s)(x_k^s - w_{jk}), \tag{7}$$

where β is a learning rate and set to unity for all experiments discussed below. The Gaussian width is changed according to the equation:

$$\sigma_t = a_0 \left(\frac{0.005}{a_0} \right)^{\frac{t}{T}} + \frac{1}{a_0}, \tag{8}$$

where a_0 is the half of the network size, t is the number of epochs and T is the total number of epochs. The term $1/a_0$ is introduced to stabilize learning.

3 Results and Discussion

In the following experiments, we try to show how well the new method extracts features in input patterns. For easy comparison, we use the conventional SOM[1]. All data in the following experiments were normalized, and the variance was unity or its range was between zero and one.

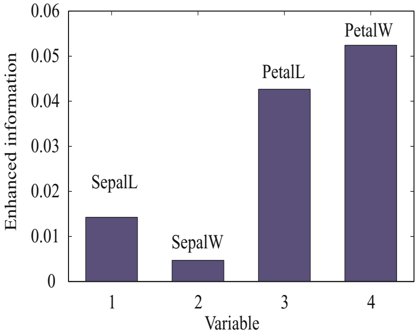
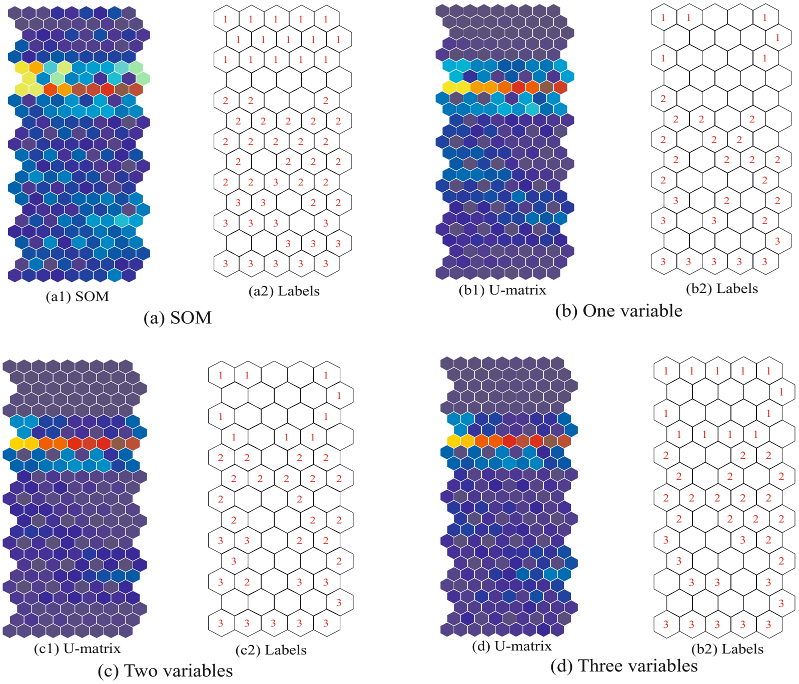


Fig. 2. Enhanced information for four variables

3.1 Iris Problem

We applied the method to the famous Iris problem for easy comparison. Figure 2 shows enhanced information for four variables. We can see that, compared with the first and the second enhanced information, the fourth and third variables show large enhanced information. Figure 3(a) shows a U-matrix and labels obtained by SOM. In the U-matrix, a clear boundary can be detected. When partial enhancement is applied with one variable (b) to three variables (c), the boundary

[1] We used SOM Toolbox 2.0, February 11th, 2000 by Juha Vesanto http://www.cis.hut.fi/projects/somtoolbox/. No special options were used for easy reproduction.


(a1) SOM (a2) Labels

(a) SOM

(b1) U-matrix (b2) Labels

(b) One variable

(c1) U-matrix (c2) Labels

(c) Two variables

(d) U-matrix (b2) Labels

(d) Three variables

Fig. 3. U-matrix and labels obtained by SOM (a) and by partial enhancement (b)-(d). Warmer and cooler colors represent larger and smaller values, respectively.

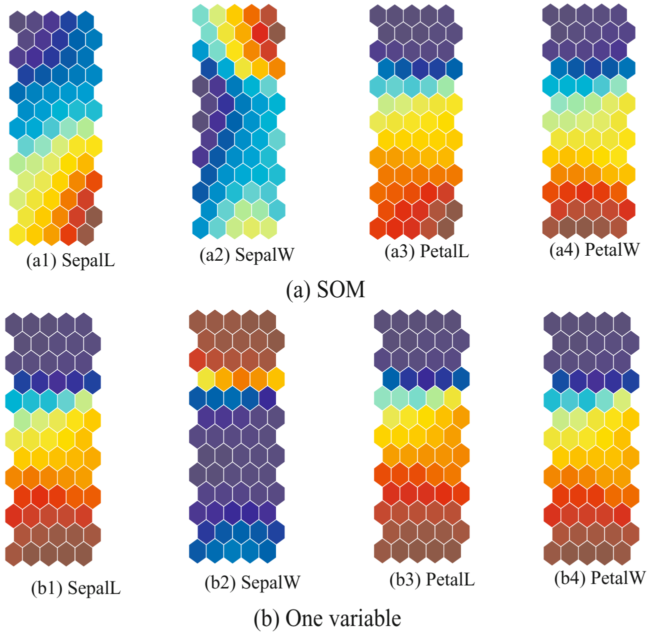
becomes clearer. Figure 4 shows component planes obtained by SOM (a) and by partial enhancement (b). For the purpose of page limitation, only component planes with one variable are shown. We can see that, with one-variable enhancement, component planes become clearer. In particular, when we see variables No. 1 and No. 2 in Figure 4 (a) and (b), obtained component planes become more similar to the U-matrix in Figure 3.

3.2 Air Pollution Problem

We tried to examine how much of polluted substances countries discharge[2]. The substances are composed of four types, that is, *sulfur*, *nitrogen*, *carbon-monoxide* and *non-methane*. The quantity is measured in kilograms per GDP. Figure 5 shows enhanced information for input units. As can be seen in the figure, variables No. 1 and No. 3 show higher enhanced information and should play important roles. That means that the substances *sulfur* and *carbon-monoxide* are critical in classification.

Figure 6 shows U-matrices and labels obtained by SOM (a) and partial activations (b)-(d). Figure 6(a) shows a U-matrix (a1) and label (a2) obtained by

[2] http://www.e-stat.go.jp/SG1/estat/eStatTopPortal.do.



(a) SOM

(b) One variable

Fig. 4. Component planes obtained by SOM (a) and by partial enhancement (b)

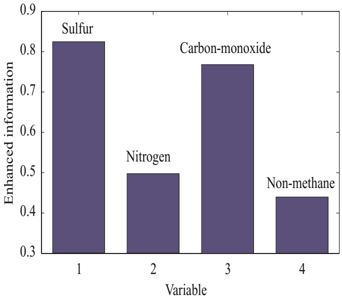
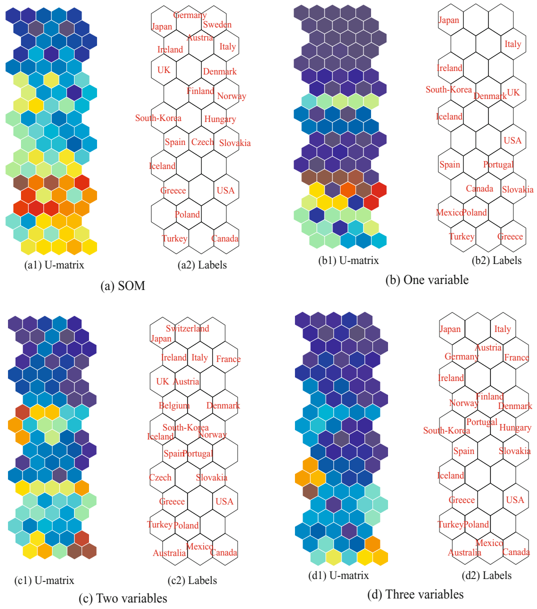


Fig. 5. Enhanced information for four variables

conventional SOM. As can be seen in the figure, clear boundaries in red or brown can be seen on the lower side of the map; however, the boundaries are rather wide. In other words, complicated boundaries can be seen on the lower side of the U-matrix. On the other hand, Figure 6(b) shows a U-matrix and labels after only one-variable enhancement. We can see two clear boundaries. One is slightly weak, in light green, and located in the middle of the map. The other one is strong, in red and brown, and it is located on the lower side of the map. As the number of variables is increased from two to three, in Figure 6(c)-(d), these boundaries become obscure. However, we can still trace the boundaries on the maps.



(a1) U-matrix (a2) Labels

(a) SOM

(b1) U-matrix (b2) Labels

(b) One variable

(c1) U-matrix (c2) Labels

(c) Two variables

(d1) U-matrix (d2) Labels

(d) Three variables

Fig. 6. U-matrices and labels obtained by conventional SOM (a) and by enhancement (b)-(d)

4 Conclusion

In this paper, we have proposed a new method to extract explicit feature maps. In this model, competitive units are partially enhanced by using some specific variables. Networks are retrained to imitate partially enhanced states. First, networks are trained with a conventional method such as SOM. Then, enhanced information for each variable is detected. According to the magnitude of enhanced information, the relevance of variables is determined. Then, with partial enhancement, for example, of one variable, networks are retrained. We applied the method to two problems: the Iris problem and the air pollution problem. In both problems we succeeded in extracting salient features in input patterns. We have confined ourselves to partial enhancement by variables. However, any other elements, such as input patterns and competitive units, can be used to enhance competitive units. Thus, we should examine the performance of partial enhancement through other elements in learning.

References

1. Versanto, J.: SOM-based data visualization methods. Intelligent Data Analysis 3, 111–126 (1999)
2. Kaski, S., Nikkila, J., Kohonen, T.: Methods for interpreting a self-organized map in data analysis. In: Proceedings of European Symposium on Artificial Neural Networks, Bruges, Belgium (1998)

3. Mao, I., Jain, A.K.: Artificial neural networks for feature extraction and multivariate data projection. IEEE Transactions on Neural Networks 6(2), 296–317 (1995)
4. Ultsch, A., Siemon, H.P.: Kohonen self-organization feature maps for exploratory data analysis. In: Proceedings of International Neural Network Conference, pp. 305–308. Kluwer Academic Publisher, Dordrecht (1990)
5. Kamimura, R.: Teacher-directed learning: information-theoretic competitive learning in supervised multi-layered networks. Connection Science 15, 117–140 (2003)
6. Kamimura, R.: Information-theoretic competitive learning with inverse euclidean distance. Neural Processing Letters 18, 163–184 (2003)
7. Kamimura, R.: Improving information-theoretic competitive learning by accentuated information maximization. International Journal of General Systems 34(3), 219–233 (2006)
8. Kamimura, R.: Free energy-based competitive learning for mutual information maximization. In: Proceedings of IEEE Conference on Systems, Man, and Cybernetics, pp. 223–227 (2008)
9. Kamimura, R.: Free energy-based competitive learning for self-organizing maps. In: Proceedings of Artificial Intelligence and Applications, pp. 414–419 (2008)

Feature Detection by Structural Enhanced Information

Ryotaro Kamimura

IT Education Center, Tokai University
1117 Kitakaname Hiratsuka Kanagawa 259-1292, Japan
ryo@cc.u-tokai.ac.jp

Abstract. In this paper, we propose structural enhanced information for detecting main features in input patterns. In structural enhanced information, three types of enhanced information can be differentiated, that is, the first-, the second- and the third-order enhanced information. The first-order information is related to the enhancement of competitive units themselves through some elements in a network, and the second-order information is dependent upon the enhancement of competitive units with input patterns. Then, the third-order information is obtained by subtracting the effect of the first-order information from the second-order information. Thus, the third-order information more explicitly represents information on input patterns. With this structural enhanced information, we can estimate more detailed features in input patterns. We applied the method to the well-known Iris problem. In both problems, we succeeded in extracting detailed and important features especially by using the third-order information.

1 Introduction

Much attention has been paid to feature extraction and discovery in neural networks. For example, for supervised learning, sensitivity analysis [1], [2], [3] has been used to interpret final representations after learning. One of the major problems of neural networks consists in difficulty in interpreting final internal representations. However, to our best knowledge, due attention has not been paid to this problem for unsupervised learning, though much effort has been made to improve classification performance in competitive learning [4], [5], [6], [7], [8] and visualization performance [9], [10], [11] in self-organizing maps. This is because it has been difficult to identify criteria comparable to those in errors terms between targets and outputs. In this context, we have introduced the information-theoretic approach to feature detection.

Information-theoretical methods have been proposed to describe many aspects of neural computing [12], [13], [14]. We have, especially, pointed out the similarity between information-theoretic learning and competitive learning [15], [16]. In addition to the similarity between it and competitive learning, the information-theoretic learning method has presented a solution for the fundamental problems of competitive learning, such as the dead neuron problem [17], [4], [5], [8].

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 171–178, 2009.

However, as an information measure, mutual information used in information-theoretic formulation is only an average of all input patterns and competitive units. It does not give useful information on detailed parts in neural networks.

To obtain useful information on detailed parts of a network, we introduce *enhancement* and *relaxation* for competitive units. By the enhancement of a competitive unit, the unit responds explicitly to all the input patterns, while by relaxation of the competitive unit, the unit responds uniformly to all the input patterns. Information change by enhancement or relaxation is called *enhanced information*. Enhanced information seems to be a powerful method to detect features. However, we have found that, if we consider several types of enhanced information, we can more clearly detect important features in input patterns. In this context, we introduce structural enhanced information in which the first-, the second- and the third-order enhanced information are differentiated. The first-order information is an information change due only to competitive units. On the other hand, the second-order information is an information change due to competitive units with input patterns. The third-order information is obtained by subtracting the effect of the first-order information from the second-order information. The third-order information is used to produce information without the effect of the first-order information. Because we take into account the structure of information content, this information is called *structural enhanced information*. Though all types of information play important roles in interpreting final representations, we focus upon the third-order information in this paper and try to show that the third-order information is good at extracting important features in input patterns.

2 Theory and Computational Methods

2.1 Enhancement and Relaxation

Figure 1 shows a process of enhancement and relaxation. Figure 1(a) shows an original situation obtained by competitive learning, in which three neurons are differently activated for input units. With enhancement, as shown in Figure 1(b), the characteristics of competitive unit activations are enhanced, and only one competitive unit is strongly activated. This means that obtained information in competitive units is larger. On the other hand, Figure 1(c) shows a state by relaxation, in which all competitive units respond uniformly to input units. Because competitive units cannot differentiate between input patterns, no information on input patterns is stored.

This enhancement can be realized by changing the Gaussian width. We consider a network for competitive learning, shown in Figure 2, where x_k^s denotes the kth element of the sth input pattern, and w_{jk} represent connection weights from the kth input unit to the jth competitive unit. In this network, a neuron output can be defined by the Gaussian function:

$$v_j^s = \exp\left(-\frac{\sum_{k=1}^{L}(x_k^s - w_{jk})^2}{2\sigma^2}\right), \qquad (1)$$

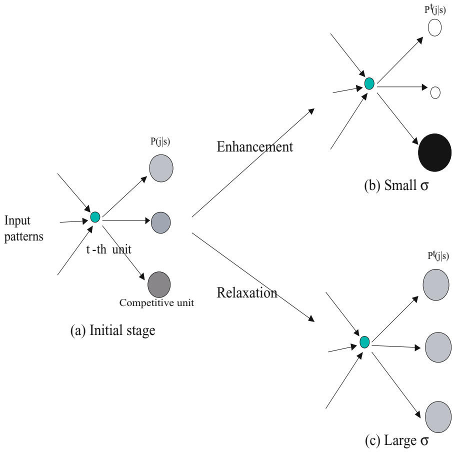


Fig. 1. Enhancement (b) and relaxation (c) in competitive learning

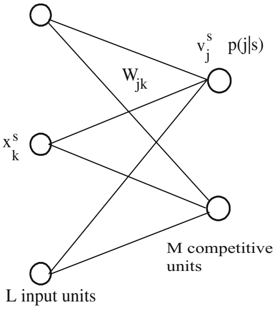


Fig. 2. A network architecture for competition

where σ is a Gaussian width and L is the number of input units. As the Gaussian width is smaller, competitive units are considered to be more enhanced.

2.2 Structural Enhanced Information

In this paper, enhancement is measured in three different ways in order to see a more detailed change in neurons. For this purpose, we introduce structural enhanced information, in which we can define three types of enhanced information. Let us introduce structural enhanced information. Suppose that the probability $p(j)$ of firing of the jth neuron is changed to $p^{enh}(j)$ by enhancement, and the conditional probability $p(j \mid s)$ to $p^{enh}(j \mid s)$. The first-order enhanced information for the probability $p(j)$ can be defined by using the cross entropy:

$$EI_1 = \sum_{j=1}^{M} p(j) \log \frac{p(j)}{p^{enh}(j)}. \tag{2}$$

The second-order enhanced information is defined by

$$EI_2 = \sum_{s=1}^{S} p(s) \sum_{j=1}^{M} p(j \mid s) \log \frac{p(j \mid s)}{p^{enh}(j \mid s)}. \quad (3)$$

By an analogy with mutual information, we have the third-order enhanced information

$$\begin{aligned} EI_3 &= EI_2 - EI_1 \\ &= \sum_{s=1}^{S} p(s) \sum_{j=1}^{M} p(j \mid s) \log \frac{p(j \mid s)}{p^{enh}(j \mid s)} - \sum_{j=1}^{M} p(j) \log \frac{p(j)}{p^{enh}(j)}. \end{aligned} \quad (4)$$

This third-order enhanced information is formulated by our expectation of the second-order enhanced information to be much larger than the first-order information. The first-order enhanced information is concerned with information change due to competitive units themselves. On the other hand, the second-order enhanced information is related to information change due to competitive units with input patterns. The third-order enhanced information is information change due to competitive units with input patterns without the effect due to the competitive units themselves.

2.3 Structural Enhanced Information for Input and Competitive Units

We define only enhanced information for input and competitive units with page limitation. However, other types of enhanced information can easily be obtained in the same way. Distance between input units and connection weights when the rth competitive unit and the kth input unit are used for enhancement are given by

$$d^s_{kt,jr} = \sum_{k=1}^{L} \Phi_{kt,jr} (x^s_k - w_{jk})^2, \quad (5)$$

where

$$\Phi_{kt,jr} = \begin{cases} 1/\epsilon, & \text{if } k = t \text{ and } j = r; \\ \epsilon, & \text{otherwise,} \end{cases}$$

where

$$\epsilon = \frac{1}{2\alpha\sigma^2}. \quad (6)$$

By using this distance, we have competitive units activations through the kth unit and the jth unit:

$$v^s_{kt,jr} = \exp\left(-\frac{d^s_{kt,jr}}{2\sigma^2}\right). \quad (7)$$

We can also define the first order enhanced information

$$EI_1^{tr} = \sum_{j=1}^{M} p(j) \log \frac{p(j)}{p^{tr}(j)}. \tag{8}$$

By using these probabilities, we have the second order enhanced information for the rth competitive unit

$$EI_2^{tr} = \sum_{s=1}^{S} \sum_{j=1}^{M} p(s)p(j \mid s) \log \frac{p(j \mid s)}{p^{tr}(j \mid s)}. \tag{9}$$

The third order enhanced information is defined by difference between two enhanced information defined by

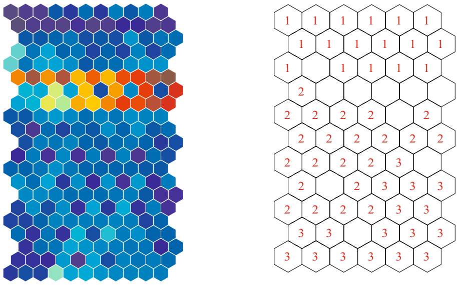
$$EI_3^{tr} = EI_2^{tr} - EI_1^{tr}. \tag{10}$$

2.4 Iris Problem

In this experiment, we use the well-known Iris problem to show how well the new method extracts features in input patterns. The data is normalized and the variance is unity. The number of competitive units is 66 (6 by 11), specified by the SOM software package[1].

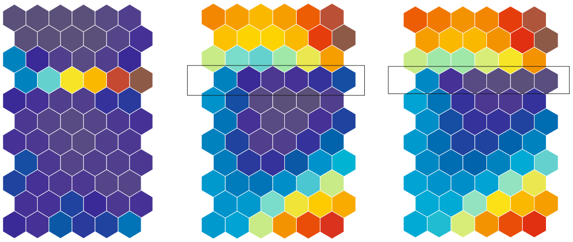
Figure 3 shows a U-matrix (a) and a map with labels (b). As can be seen in Figure 3(a), the U-matrix shows a clear boundary in red or brown by which input patterns can be classified into two groups. However, in the U-matrix we cannot see a boundary between class No. 2 and class No. 3. The boundary between classes No. 2 and No. 3 can be found in the map with labels, shown in Figure 3(b). Figure 4 shows the first-order enhanced information (a), the second-order information (b) and the third-order information (c). As shown in Figure 4(a), the first-order enhanced information for competitive units clearly detects a boundary in warmer colors, such as brown and red, in the middle, corresponding to the boundary obtained by the SOM in Figure 3(a). This means that, with first-order enhanced information, input patterns are classified into two groups. Figure 4(b) shows the second-order enhanced information, which shows three groups on the map. The first group is located on the upper end of the map in red and brown. In the middle, we can see a group in cooler colors. Finally, on the lower end, we can see another group in warmer colors. Figure 4(c) shows the third-order enhanced information. Though it is slightly difficult to see details on the map, we can say that the third-order enhanced information shows a boundary in darker blue in the middle that is clearer or darker than that obtained by the second-order information in Figure 4(b). Competitive units corresponding to a boundary in Figure 3(a) are represented in darker blue, meaning smaller values. This effect is obtained by subtracting the first-order information from the second-order information.

[1] http://www.cis.hut.fi/research/som-research.



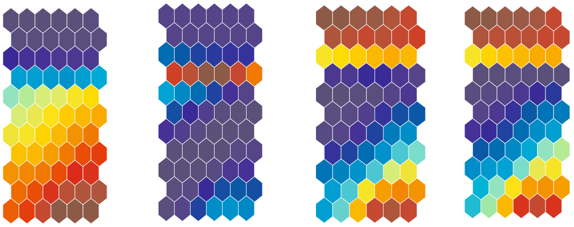
(a) U-matrix (b) Labels

Fig. 3. U-matrix (a) and a map with labels (b). Warmer and cooler colors represent larger and smaller values, respectively.



(a) First order (b) Second order (c) Third order

Fig. 4. The first-order enhanced information EI_1^r (a), the second-order information EI_2^r (b) and the third-order information EI_3^r (c). Warmer and cooler colors represent larger and smaller values, respectively. Frames in the figures show that competitive units in the frame in (c) are smaller (darker) than that in (b). Warmer and cooler colors represent larger and smaller values, respectively.



(a) Component planes (b) First-order (c) Second order (d) Third order planes

Fig. 5. Component planes (a) and the first-order (b), the second-order (c) and the third-order enhanced information for input and competitive units. Warmer and cooler colors represent larger and smaller values, respectively.

Figure 5 shows component planes obtained for the variable No.3 with the highest enhanced information by SOM (a) and the first-order (b), the second-order (c) and the third-order (d) enhanced information. Figure 5(b) shows the first-order enhanced information for the variable No.3. The enhanced information (Figure 5(b)) shows a strong boundary in red and brown, corresponding to the

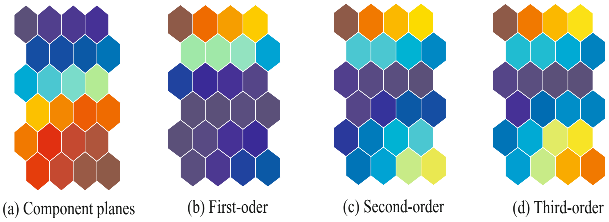


Fig. 6. Component planes (a), the first-order (b), the second-order (c) and the third-order (d) enhanced information. Warmer and cooler colors represent larger and smaller values, respectively.

boundary in Figure 3(a). The enhanced information of the second and third order in Figure 5(c) and (d) are similar to each other. Though it is relatively difficult to see the details, we can see that the third-order enhanced information is one where a boundary in the second-order information in Figure 5(c) is accentuated.

3 Conclusion

In this paper, we have introduced a new type of information called *enhanced information* as well as *structural enhanced information*. Enhanced information is obtained by enhancing competitive units through some elements in a network, while all the other competitive units are forced to be relaxed. If this enhancement causes a drastic change in information for competitive units, the elements surely play a very important role in information processing in competitive learning. To see in more detail the role of this enhanced information, we have introduced structural enhanced information, with three types of information, that is, first, second- and the third-order enhanced information. The first-order enhanced information consists of change only in competitive units. The second-order enhanced information consists of change of competitive units, given input patterns. The third-order enhanced information is the difference between the first- and the second-order enhanced information. We have applied the method to the well-known Iris problem to show easily how well the new method discovers features in input patterns. Experimental results have shown that features extracted by the new method are clearer than those extracted by the conventional SOM, and results correspond to our intuition on input patterns. Though much effort remains for the method to be made practically applicable, it is certain that this study opens up a new perspective in the information-theoretic approach to unsupervised learning.

References

1. Mozer, M.C., Smolensky, P.: Using relevance to reduce network size automatically. Connection Science 1(1), 3–16 (1989)
2. Karnin, E.D.: A simple procedure for pruning back-propagation trained networks. IEEE transactions on neural networks 1(2), 239–242 (1990)

3. Reed, R.: Pruning algorithms-a survey. IEEE Transactions on Neural Networks 4(5) (1993)
4. DeSieno, D.: Adding a conscience to competitive learning. In: Proceedings of IEEE International Conference on Neural Networks, San Diego, pp. 117–124. IEEE, Los Alamitos (1988)
5. Ahalt, S.C., Krishnamurthy, A.K., Chen, P., Melton, D.E.: Competitive learning algorithms for vector quantization. Neural Networks 3, 277–290 (1990)
6. Xu, L.: Rival penalized competitive learning for clustering analysis, RBF net, and curve detection. IEEE Transaction on Neural Networks 4(4), 636–649 (1993)
7. Luk, A., Lien, S.: Properties of the generalized lotto-type competitive learning. In: Proceedings of International conference on neural information processing, San Mateo, CA, pp. 1180–1185. Morgan Kaufmann Publishers, San Francisco (2000)
8. Hulle, M.M.V.: The formation of topographic maps that maximize the average mutual information of the output responses to noiseless input signals. Neural Computation 9(3), 595–606 (1997)
9. Kaski, S., Nikkilä, J., Kohonen, T.: Methods for interpreting a self-organized map in data analysis. In: Verleysen, M. (ed.) Proceedings of ESANN 1998, 6th European Symposium on Artificial Neural Networks, Bruges, April 22–24, 1998, pp. 185–190. D-Facto, Brussels (1998)
10. Vesanto, J., Alhoniemi, E.: Clustering of the self-organizing map. IEEE-NN 11, 586 (2000)
11. Vesanto, J.: SOM-based data visualization methods. Intelligent-Data-Analysis 3, 111–126 (1999)
12. Torkkola, K.: Feature extraction by non-parametric mutual information maximization. Journal of Machine Learning Research 3, 1415–1438 (2003)
13. Slonim, N., Tishby, N.: Agglomerative information bottleneck (1999)
14. Linsker, R.: Self-organization in a perceptual network. Computer 21, 105–117 (1988)
15. Kamimura, R., Kamimura, T., Uchida, O.: Flexible feature discovery and structural information. Connection Science 13(4), 323–347 (2001)
16. Kamimura, R.: Information-theoretic competitive learning with inverse euclidean distance. Neural Processing Letters 18, 163–184 (2003)
17. Rumelhart, D.E., Zipser, D.: Feature discovery by competitive learning. Cognitive Science 9, 75–112 (1985)

Gradient Learning in Networks of Smoothly Spiking Neurons

Jiří Šíma[⋆]

Institute of Computer Science, Academy of Sciences of the Czech Republic,
P. O. Box 5, 18207 Prague 8, Czech Republic
sima@cs.cas.cz

Abstract. A slightly simplified version of the Spike Response Model SRM_0 of a spiking neuron is tailored to gradient learning. In particular, the evolution of spike trains along the weight and delay parameter trajectories is made perfectly smooth. For this model a back-propagation-like learning rule is derived which propagates the error also along the time axis. This approach overcomes the difficulties with the discontinuous-in-time nature of spiking neurons, which encounter previous gradient learning algorithms (e.g. *SpikeProp*). The new algorithm can naturally cope with multiple spikes and preliminary experiments confirm the smoothness of spike creation/deletion process.

1 Learning in Networks of Spiking Neurons

The prominent position among neural network models has recently been occupied by networks of spiking (pulse) neurons [4]. As compared to the traditional perceptrons the spiking neurons represent a biologically more plausible model which has a great potential for processing the temporal information. However, one of the main open issues is the development of a practical learning algorithm for the networks of spiking neurons although the related training problem is known to be NP-complete at least for binary coded data [6,11].

Learning in the perceptron networks is usually performed by a gradient descent method, e.g. by using the back-propagation algorithm which explicitly evaluates the gradient of an error function. For the first time, the same approach was employed for spiking networks in the *SpikeProp* algorithm [2] which learns the desired firing times of output neurons by adapting the weight parameters in the Spike Response Model SRM_0 [4]. Plenty of experiments with *SpikeProp* was carried out which clarified e.g. the role of the parameter initialization and negative weights [7]. The performance of the original algorithm was improved by adding the momentum term [14]. Moreover, the *SpikeProp* was further enhanced with additional learning rules for synaptic delays, thresholds, and time constants [8] which resulted in faster convergence and smaller network sizes for

[⋆] Research partially supported by projects 1ET100300517 (AS CR), 1M0545 (MŠMT CR), and AV0Z10300504 (ICS AS CR). The author also thanks Petra Vídnerová for her help with computer experiments.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 179–186, 2009.

given learning tasks. An essential speedup was achieved by approximating the firing time function using the logistic sigmoid [1]. The *SpikeProp* algorithm was also extended to recurrent network architectures [13].

Nevertheless, the *SpikeProp* method and its modifications do not usually allow more than one spike per neuron which makes it suitable only for 'time-to-first-spike' coding scheme [12]. Another important drawback of these learning heuristics is that the adaptation mechanism fails for the weights of neurons that do not emit spikes. At the core of these difficulties the fact is that the spike creation or removal due to weight updates is by nature a very discontinuous process while in contrast the gradient learning algorithms do essentially require the error function to be perfectly smooth for their implementations. Recently, the so-called *ASNA-Prop* has been proposed [9] to solve this problem by emulating the feedforward networks of spiking neurons with the discrete-time analog sigmoid networks with local feedback, which is then used for deriving the gradient learning rule. Another method estimates the gradient by measuring the fluctuations in the error function in response to the dynamic neuron parameter perturbation [3]. See also paper [5] for a recent review of supervised learning methods in the spiking neural networks.

In the present paper we employ a different approach to this problem of spike creation/deletion. Similarly as the Heaviside function was replaced by the logistic sigmoid in the conventional back-propagation algorithm to make the neuron function differentiable, we modify a slightly simplified version of the Spike Response Model SRM_0 by smoothing out the discontinuities also along the weight and delay parameter trajectories (Section 2). Thus, a new spike arises through a continuous "division" of the preceding spike into two spikes while the spike disappearance is implemented by a continuous "fusion" with its predecessor. For our model of smoothly spiking neurons we derive a nontrivial back-propagation-like rule for computing the gradient of the error function with respect to both the weight and delay parameters which is only sketched in Section 3 due to lack of space while the complete formulas can be found in our technical report [10]. In particular, the chain rule for computing the partial derivatives of the composite error function is implemented so that each neuron propagates backwards a variable number of partial derivative terms corresponding to different time instants including the second-order derivative terms. Thus, the new gradient learning method can naturally cope with multiple spikes whose number changes in time and can be used for supervised learning of desired spike trains in the networks of spiking neurons. Preliminary experiments with the proposed learning algorithm exhibit the smoothness of spike creation/deletion process (Section 4).

2 A Feedforward Network of Smoothly Spiking Neurons

Formally, a feedforward network can be defined as a set of *spiking (pulse) neurons* V which are densely connected into a *directed* (connected) graph representing an *architecture* of the network. Some of these neurons may serve as external inputs or outputs, and hence we assume $X \subseteq V$ and $Y \subseteq V$ to be a set of *input* and *output* neurons, respectively, while the remaining ones are called *hidden* neurons. We denote by $j_{\leftarrow}$ the set of all neurons from which an edge leads to j while $j^{\rightarrow}$ denotes the set of all neurons to which an edge leads from j. As usual we assume $j_{\leftarrow} = \emptyset$ for $j \in X$ and $j^{\rightarrow} = \emptyset$ for $j \in Y$ whereas $j_{\leftarrow} \neq \emptyset$ and $j^{\rightarrow} \neq \emptyset$ for $j \in V \setminus (X \cup Y)$. Each edge in the architecture leading from neuron i to j is labeled with a real *(synaptic) weight* w_{ji} and *delay* $d_{ji} \geq 0$. In addition, a real *bias* parameter w_{j0} is assigned to each noninput neuron $j \in V \setminus X$ which can be viewed as the weight from a formal constant unit input[1]. All these weights and delays altogether create the network parameter vectors $\mathbf{w}$ and $\mathbf{d}$, respectively.

We first define an auxiliary function that will be used for making the computational dynamics of the network smooth with respect to both the time and parameters $\mathbf{w}$ and $\mathbf{d}$. In particular, a twice differentiable nondecreasing function of one variable $\sigma(\alpha, \beta, \delta) : \Re \longrightarrow [\alpha, \beta]$ (or σ for short) is introduced which has three real parameters $\alpha \leq \beta$ and $\delta > 0$ such that $\sigma(x) = \alpha$ for $x \leq 0$ and $\sigma(x) = \beta$ for $x \geq \delta$ whereas the first and second derivatives satisfy $\sigma'(0) = \sigma'(\delta) = \sigma''(0) = \sigma''(\delta) = 0$. In fact, σ is a smooth approximation of the step function where δ controls the approximation error. In order to fulfill the conditions on the derivatives we can employ, e.g., the primitive function $\int Ax^2(x-\delta)^2\,dx = A\left(\frac{x^5}{5} - 2\delta\,\frac{x^4}{4} + \delta^2\,\frac{x^3}{3}\right) + C$ for a normalized $\sigma_0 = \sigma(0, 1, \delta)$ on $[0, \delta]$ where the real constants $C = 0$ and $A = 30/\delta^5$ are determined from $\sigma_0(0) = 0$ and $\sigma_0(\delta) = 1$. Thus we can choose $\sigma(\alpha, \beta, \delta\,;\, x) = (\beta - \alpha)\,\sigma_0(x) + \alpha$ for $x \in [0, \delta]$ which results in the following definition:

$$\sigma(\alpha, \beta, \delta\,;\, x) = \begin{cases} \alpha & \text{for } x < 0 \\ (\beta - \alpha)\left(\left(6\frac{x}{\delta} - 15\right)\frac{x}{\delta} + 10\right)\left(\frac{x}{\delta}\right)^3 + \alpha & \text{for } 0 \leq x \leq \delta \\ \beta & \text{for } x > \delta\,. \end{cases} \tag{1}$$

We will also use the logistic sigmoid function with a gain parameter λ (e.g. $\lambda = 4$):

$$P(x) = \frac{1}{1 + e^{-\lambda x}}\,. \tag{2}$$

Now we introduce the smooth computational dynamics. Each spiking neuron $j \in V$ in the network may produce a sequence of p_j *spikes (firing times)* $0 < t_{j1} < t_{j2} < \cdots < t_{jp_j} < T$. In addition, define formally $t_{j0} = 0$ and $t_{j,p_j+1} = T$. For every input neuron $j \in X$, this sequence of spikes is given externally as a global input to the network. For a noninput neuron $j \in V \setminus X$, on the other hand, the underlying firing times are computed as the time instants at which its *excitation*

$$\xi_j(t) = w_{j0} + \sum_{i \in j_{\leftarrow}} w_{ji}\,\varepsilon\left(t - d_{ji} - \tau_i(t - d_{ji})\right) \tag{3}$$

evolving in time $t \in [0, T]$ crosses 0 from below, that is,

$$\left\{0 \leq t \leq T \,|\, \xi_j(t) = 0 \,\&\, \xi_j'(t) > 0\right\} = \left\{t_{j1} < t_{j2} < \cdots < t_{jp_j}\right\}\,. \tag{4}$$

[1] For simplicity, we assume a *constant* threshold function (which equals the opposite value of the bias), thus excluding the refractory effects [4].

Formula (3) defines the excitation of neuron $j \in V \setminus X$ at time t as a weighted sum of responses to the last spikes from neurons $i \in j_{\leftarrow}$ delayed by d_{ji} preceding time instant t. Here $\varepsilon : \Re \longrightarrow \Re$ denotes the smooth *response function* which is the same for all neurons, e.g.

$$\varepsilon(t) = e^{-(t-1)^2} \cdot \sigma_0(t) \qquad \text{where} \qquad \sigma_0(t) = \sigma(0, 1, \delta_0 \,;\, t) \tag{5}$$

is defined by (1) using parameter δ_0 particularly for the definition of ε. Clearly, ε is a twice differentiable function with a relatively flat spike shape which reaches its maximum $\varepsilon(1) = 1$ for $t = 1$. Furthermore, $\tau_i : \Re \longrightarrow \Re$ is a smooth approximation of the stair function that gives the last firing time t_{is} of neuron i preceding a given time t, that is $t_{is} < t \leq t_{i,s+1}$.

In particular, we define the function τ_j for any neuron $j \in V$ as follows. The firing times $t_{j1} < t_{j2} < \cdots < t_{jp_j}$ of j are first transformed one by one into $\widetilde{t_{j0}} = 0 < \widetilde{t_{j1}} < \widetilde{t_{j2}} < \cdots < \widetilde{t_{jp_j}} < T = t_{j,p_j+1}$ by formula

$$\widetilde{t_{js}} = \begin{cases} t_{js} & \text{for } j \in X \\ \sigma\left(\widetilde{t_{j,s-1}}, t_{js}, \delta \,;\, \xi_j'(t_{js})\right) & \text{for } j \in V \setminus X \end{cases} \tag{6}$$

using (1) where s goes in sequence from 1 to p_j and $\xi_j'(t_{js})$ for $j \in V \setminus X$ is the derivative of excitation ξ_j at time t_{js} which is positive according to (4). Clearly, $\widetilde{t_{js}} = t_{js}$ for $\xi_j'(t_{js}) \geq \delta$ while $\widetilde{t_{js}} \in (\widetilde{t_{j,s-1}}, t_{js})$ is smoothly growing with increasing small $\xi_j'(t_{js}) \in (0, \delta)$. The purpose of this transformation is to make the creation and deletion of spikes smooth with respect to the weight and delay updates. In particular, by moving along the weight and delay trajectories, the excitation ξ_j may reach and further cross 0 from below (spike creation), which leads to an increase in the first derivative of excitation $\xi_j'(t_{js})$ at the crossing point t_{js} starting from zero. Thus, the new transformed spike $\widetilde{t_{js}}$ arises through a continuous "division" of the preceding transformed spike $\widetilde{t_{j,s-1}}$ into two spikes while the spike disappearance is implemented similarly by a continuous "fusion" with its predecessor, which is controlled by the first derivative of the excitation. The transformed spikes then define

$$\tau_j(t) = \sum_{s=1}^{p_j+1} \left(\widetilde{t_{js}} - \widetilde{t_{j,s-1}}\right) P^c\left(t - \widetilde{t_{js}}\right) \tag{7}$$

using the logistic sigmoid (2) where $c \geq 1$ (e.g. $c = 3$) is an optional exponent whose growth decreases the value of $P(0)$ shifting the transient phase of $P(x)$ to positive x. Finally, the derivative of j's excitation with respect to time t is calculated as

$$\xi_j'(t) = \sum_{i \in j_{\leftarrow}} w_{ji}\, \varepsilon'\left(t - d_{ji} - \tau_i(t - d_{ji})\right)\left(1 - \tau_i'(t - d_{ji})\right) \tag{8}$$

where $\varepsilon'(t) = e^{-(t-1)^2} \cdot (\sigma_0'(t) - 2(t-1)\sigma_0(t))$ and

$$\tau_i'(t) = \frac{\partial}{\partial\, t}\tau_i(t) = c\lambda \sum_{s=1}^{p_i+1} \left(\widetilde{t_{is}} - \widetilde{t_{i,s-1}}\right) P^c\left(t - \widetilde{t_{is}}\right)\left(1 - P\left(t - \widetilde{t_{is}}\right)\right) . \tag{9}$$

3 The Back-Propagation Rule

A *training pattern* associates a global temporal input specifying the spike trains $0 < t_{i1} < t_{i2} < \cdots < t_{ip_i} < T$ for every input neuron $i \in X$ with corresponding sequences of desired firing times $0 < \varrho_{j1} < \varrho_{j2} < \cdots < \varrho_{jq_j} < T$ prescribed for all output neurons $j \in Y$. The discrepancy between the desired and actual sequences of spikes for the underlying global temporal input can be measured by the following L_2 *error*

$$E(\mathbf{w}, \mathbf{d}) = \frac{1}{2} \sum_{j \in Y} \sum_{s=0}^{q_j} \left(\tau_j \left(\varrho_{j,s+1} \right) - \varrho_{js} \right)^2 \tag{10}$$

which is a function of the weight and delay parameters $\mathbf{w}$ and $\mathbf{d}$, respectively. In particular, $\tau_j(\varrho_{j,s+1})$ is the smooth approximation of the last firing time of output neuron $j \in Y$ preceding time instant $\varrho_{j,s+1}$ which is desired to be ϱ_{js}.

We will derive a back-propagation-like rule for computing the gradient of the error function (10) which can then be minimized using a gradient descent method, e.g. for every $j \in V$ and $i \in j_{\leftarrow}$ (or $i = 0$ for bias w_{j0})

$$w_{ji}^{(t)} = w_{ji}^{(t-1)} - \alpha \frac{\partial E}{\partial w_{ji}} \left(\mathbf{w}^{(t-1)} \right) , \quad d_{ji}^{(t)} = d_{ji}^{(t-1)} - \alpha \frac{\partial E}{\partial d_{ji}} \left(\mathbf{d}^{(t-1)} \right) \tag{11}$$

starting with suitable initial paramater values $\mathbf{w}^{(0)}, \mathbf{d}^{(0)}$ where $0 < \alpha < 1$ is a learning rate. Unlike the conventional back-propagation learning algorithm for the perceptron network, the new rule for the network of spiking neurons must take also the temporal dimension into account. In particular, the chain rule for computing the partial derivatives of the composite error function $E(\mathbf{w}, \mathbf{d})$ requires each neuron to propagate backwards a certain number of partial derivative terms corresponding to different time instants. For this purpose, each noninput neuron $j \in V \setminus X$ stores a list P_j of m_j ordered triples $(\pi_{jc}, \pi'_{jc}, u_{jc})$ for $c = 1, \ldots, m_j$ where π_{jc}, π'_{jc} denote the values of derivative terms associated with time u_{jc} such that

$$\frac{\partial E}{\partial w_{ji}} = \sum_{c=1}^{m_j} \left(\pi_{jc} \cdot \frac{\partial}{\partial w_{ji}} \tau_j(u_{jc}) + \pi'_{jc} \cdot \frac{\partial}{\partial w_{ji}} \tau'_j(u_{jc}) \right) \text{ for } i \in j_{\leftarrow} \cup \{0\} \tag{12}$$

(similarly for $\frac{\partial E}{\partial d_{ji}}$). Notice that the triples $(\pi_{jc_1}, \pi'_{jc_1}, u_{jc_1})$ and $(\pi_{jc_2}, \pi'_{jc_2}, u_{jc_2})$ such that $u_{jc_1} = u_{jc_2}$ can be merged into one $(\pi_{jc_1} + \pi_{jc_2}, \pi'_{jc_1} + \pi'_{jc_2}, u_{jc_1})$ and also the triples $(\pi_{jc}, \pi'_{jc}, u_{jc})$ with $\pi_{jc} = \pi'_{jc} = 0$ can be omitted.

For any noninput neuron $j \in V \setminus X$ we will calculate the underlying derivative terms π_{jc}, π'_{jc} at required time instants u_{jc}. For an output neuron $j \in Y$ the list

$$P_j = ((\tau_j(\varrho_{j,s+1}) - \varrho_{js} \, , \, 0 \, , \, \varrho_{j,s+1}) \, , \, s = 0, \ldots, q_j) \tag{13}$$

is obtained directly from (10). For creating P_i for a hidden neuron $i \in j_{\leftarrow}$ for some $j \in V \setminus X$, we derive a recursive procedure using the partial derivatives

$$\frac{\partial}{\partial w_{i\ell}} \tau_j(t) = \sum_{s=1}^{p_j} \frac{\partial}{\partial \widetilde{t_{js}}} \tau_j(t) \cdot \frac{\partial \widetilde{t_{js}}}{\partial w_{i\ell}} \, , \quad \frac{\partial}{\partial w_{i\ell}} \tau'_j(t) = \sum_{s=1}^{p_j} \frac{\partial}{\partial \widetilde{t_{js}}} \tau'_j(t) \cdot \frac{\partial \widetilde{t_{js}}}{\partial w_{i\ell}} \tag{14}$$

where $\frac{\partial}{\partial \widetilde{t_{js}}}\tau_j(t)$ and $\frac{\partial}{\partial \widetilde{t_{js}}}\tau_j'(t)$ are calculated by differentiating (7) and (9), respectively. Furthermore,

$$\frac{\partial \widetilde{t_{js}}}{\partial w_{i\ell}} = \frac{\partial \widetilde{t_{js}}}{\partial \widetilde{t_{j,s-1}}} \cdot \frac{\partial \widetilde{t_{j,s-1}}}{\partial w_{i\ell}} + \left(\frac{\partial \widetilde{t_{js}}}{\partial t_{js}} \cdot \frac{\partial t_{js}}{\partial \tau_i} + \frac{\partial \widetilde{t_{js}}}{\partial \xi_j'} \cdot \frac{\partial \xi_j'}{\partial \tau_i} \right) \frac{\partial \tau_i}{\partial w_{i\ell}} \\ + \frac{\partial \widetilde{t_{js}}}{\partial \xi_j'} \cdot \frac{\partial \xi_j'}{\partial \tau_i'} \cdot \frac{\partial \tau_i'}{\partial w_{i\ell}} \qquad (15)$$

according to (6) and (8) where the particular partial derivatives can simply be calculated by differentiating the underlying formulas, e.g. using (1), except for $\frac{\partial t_{js}}{\partial \tau_i}$ whose calculation follows. According to (3) and (4), the dependence of t_{js} on τ_i can only be expressed implicitly as $\xi_j(t_{js}) = 0$ where $\xi_j'(t_{js}) \neq 0$, which implies the total differential identity

$$\xi_j'(t_{js})\, dt_{js} + \frac{\partial \xi_j}{\partial w_{i\ell}}\, dw_{i\ell} = 0 \qquad (16)$$

employing e.g. the variable $w_{i\ell}$ for which $\frac{\partial \tau_k}{\partial w_{i\ell}} = 0$ for $k \in j_{\leftarrow}$ unless $i = k$. Hence, $\xi_j'(t_{js}) \cdot \frac{\partial t_{js}}{\partial w_{i\ell}} = -\frac{\partial \xi_j}{\partial w_{i\ell}} = -\frac{\partial \xi_j}{\partial \tau_i} \cdot \frac{\partial \tau_i}{\partial w_{i\ell}}$ which gives

$$\frac{\partial t_{js}}{\partial \tau_i} = \frac{\frac{\partial t_{js}}{\partial w_{i\ell}}}{\frac{\partial \tau_i}{\partial w_{i\ell}}} = \frac{-\frac{\partial \xi_j}{\partial \tau_i}}{\xi_j'(t_{js})} = \frac{w_{ji}\, \varepsilon'(t_{js} - d_{ji} - \tau_i(t_{js} - d_{ji}))}{\xi_j'(t_{js})} . \qquad (17)$$

Now, the formula (15) for the derivative $\frac{\partial \widetilde{t_{js}}}{\partial w_{i\ell}}$ in terms of $\frac{\partial \widetilde{t_{j,s-1}}}{\partial w_{i\ell}}$ is applied recursively, which is further plugged into (14) as follows:

$$\frac{\partial}{\partial w_{i\ell}}\tau_j(t) = \sum_{s=1}^{p_j} \frac{\partial}{\partial \widetilde{t_{js}}}\tau_j(t) \sum_{r=s-n_{js}}^{s} \left(\prod_{q=r+1}^{s} \frac{\partial \widetilde{t_{jq}}}{\partial \widetilde{t_{j,q-1}}} \right) \\ \times \left(\left(\frac{\partial \widetilde{t_{jr}}}{\partial t_{jr}} \cdot \frac{\partial t_{jr}}{\partial \tau_i} + \frac{\partial \widetilde{t_{jr}}}{\partial \xi_j'} \cdot \frac{\partial \xi_j'}{\partial \tau_i} \right) \frac{\partial \tau_i}{\partial w_{i\ell}} + \frac{\partial \widetilde{t_{jr}}}{\partial \xi_j'} \cdot \frac{\partial \xi_j'}{\partial \tau_i'} \cdot \frac{\partial \tau_i'}{\partial w_{i\ell}} \right) \qquad (18)$$

(similarly for $\frac{\partial}{\partial w_{i\ell}}\tau_j'(t)$) where $1 \leq n_{js} \leq s-1$ is defined to be the least index such that $\frac{\partial \widetilde{t_{j,s-n_{js}}}}{\partial \widetilde{t_{j,s-n_{js}-1}}} = 0$ which exists since at least $\frac{\partial \widetilde{t_{j1}}}{\partial \widetilde{t_{j0}}} = 0$. Note that the product $\prod_{q=r+1}^{s} \frac{\partial \widetilde{t_{jq}}}{\partial \widetilde{t_{j,q-1}}}$ in formula (18) equals formally 1 for $r = s$. The summands of formula (18) are used for creating the list P_i for a hidden neuron $i \in V \setminus (X \cup Y)$ provided that the lists $P_j = ((\pi_{jc}, \pi_{jc}', u_{jc})\,,\, c = 1, \ldots, m_j)$ have already been created for all $j \in i^{\rightarrow}$, that is

$$P_i = \left(f_{jcsr} \left(\frac{\partial \widetilde{t_{jr}}}{\partial t_{jr}} \cdot \frac{\partial t_{jr}}{\partial \tau_i} + \frac{\partial \widetilde{t_{jr}}}{\partial \xi_j'} \cdot \frac{\partial \xi_j'}{\partial \tau_i} \right) ,\, f_{jcsr} \cdot \frac{\partial \widetilde{t_{jr}}}{\partial \xi_j'} \cdot \frac{\partial \xi_j'}{\partial \tau_i'} \,,\, t_{jr} - d_{ji} \right) \qquad (19)$$

including factors

$$f_{jcsr} = \left(\pi_{jc} \cdot \frac{\partial}{\partial\, \widetilde{t_{js}}} \tau_j(u_{jc}) + \pi'_{jc} \cdot \frac{\partial}{\partial\, \widetilde{t_{js}}} \tau'_j(u_{jc}) \right) \prod_{q=r+1}^{s} \frac{\partial\, \widetilde{t_{jq}}}{\partial\, \widetilde{t_{j,q-1}}} \qquad (20)$$

for all $j \in i^{\rightarrow}$, $c = 1, \ldots, m_j$, $s = 1, \ldots, p_j$, and $r = s - n_{js}, \ldots, s$.

Thus, the back-propagation algorithm starts with output neurons $j \in Y$ for which the lists P_j are first created by (13) and further continues by propagating the derivative terms at various time instants backwards while creating the lists P_i also for hidden neurons $i \in V \setminus (X \cup Y)$ according to (19). These lists are then used for computing the gradient of the error function (10) according to (12) where the derivatives $\frac{\partial}{\partial\, w_{ji}} \tau_j(t)$ and $\frac{\partial}{\partial\, w_{ji}} \tau'_j(t)$ are calculated analogously to (18). For example, the dependencies of t_{jr} on w_{ji} can again be expressed only implicitly as $\xi_j(t_{jr}) = 0$ and hence, the derivatives $\frac{\partial\, t_{jr}}{\partial\, w_{ji}}$ are calculated using the implicit function theorem. See [10] for more details.

4 Preliminary Experiments and Future Work

We have implemented the proposed learning algorithm and performed several preliminary computer simulations with simple toy problems such as XOR with temporal encoding. These experiments also served as a tool for debugging our model of a smoothly spiking neuron, e.g. for choosing suitable function shapes, and this work has not been finished yet. Nevertheless, the first experimental results confirm the smoothness of spike creation/deletion. For example, Figure 1 shows how the graph of function $\tau_j(t)$ evolves in the course of weight and delay adaptation for a spiking neuron j. Recall $\tau_j(t)$ is the smooth approximation of the stair function which produces the last firing time preceding a given time t. Figure 1 depicts the process of a spike creation during training when the time instant of a new spike $\widetilde{t_{js}}$ "detaches" from the preceding spike $t_{j,s-1}$ (which, for simplicity, is assumed here to be "fixed" for a moment, i.e. $\widetilde{t_{j,s-1}} = t_{j,s-1}$) and "moves" smoothly with increasing $\xi'_j(t_{js}) > 0$ to its actual position $\widetilde{t_{js}} = t_{js}$

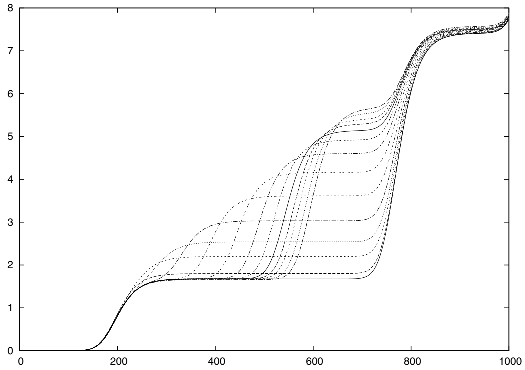


Fig. 1. Spike creation

($\xi_j(t_{js}) = 0$) where $\xi'_j(t_{js})$ reaches threshold value δ. In a general case, more spikes can smoothly be generated at the same time which was also observed in our experiments.

Nevertheless, the proposed learning algorithm for networks of smoothly spiking neurons still needs to be justified by experiments with more complicated temporal training patterns. Another challenge for further research is to generalize the model for a nonconstant threshold function taking the refractory period of a spiking neuron into account.

References

1. Berkovec, K.: Learning in networks of spiking neurons (in Czech) M.Sc. Thesis, Faculty of Mathematics and Physics, Charles University, Czech Republic (2005)
2. Bohte, S.M., Kok, J.N., La Poutré, H.: Error-backpropagation in temporally encoded networks of spiking neurons. Neurocomputing 48(1-4), 17–37 (2002)
3. Fiete, I.R., Seung, H.S.: Gradient learning in spiking neural networks by dynamic perturbation of conductances. Physical Review Letters 97, 048104 (2006)
4. Gerstner, W., Kistler, W.M.: Spiking Neuron Models: Single Neurons, Populations, Plasticity. Cambridge University Press, Cambridge (2002)
5. Kasiński, A., Ponulak, F.: Comparison of supervised learning methods for spike time coding in spiking neural networks. International Journal of Applied Mathematics and Computer Science 16(1), 101–113 (2006)
6. Maass, W., Schmitt, M.: On the complexity of learning for spiking neurons with temporal coding. Information and Computation 153(1), 26–46 (1999)
7. Moore, S.C.: Back-propagation in spiking neural networks. M.Sc. Thesis, Department of Computer Science, University of Bath, UK (2002)
8. Schrauwen, B., Van Campenhout, J.: Extending SpikeProp. In: Proceedings of the IJCNN 2004 IEEE International Joint Conference on Neural Networks, Budapest, Hungary, vol. 1, pp. 471–475. IEEE, Piscataway (2004)
9. Schrauwen, B., Van Campenhout, J.: Backpropagation for population-temporal coded spiking neural networks. In: Proceedings of the IJCNN 2006 IEEE International Joint Conference on Neural Networks, Vancouver, BC, Canada, vol. 3, pp. 3463–3470. IEEE, Piscataway (2006)
10. Šíma, J.: Gradient learning in networks of smoothly spiking neurons. Technical Report V-1045, Institute of Computer Science, Academy of Sciences of the Czech Republic, Czech Republic (2009)
11. Šíma, J., Sgall, J.: On the non-learnability of a single spiking neuron. Neural Computation 17(12), 2635–2647 (2005)
12. Thorpe, S., Delorme, A., Van Rullen, R.: Spike-based strategies for rapid processing. Neural Networks 14(6-7), 715–725 (2001)
13. Tiňo, P., Mills, A.J.S.: Learning beyond finite memory in recurrent networks of spiking neurons. Neural Computation 18(3), 591–613 (2006)
14. Xin, J., Embrechts, M.J.: Supervised learning with spiking neuron networks. In: Proceedings of the IJCNN 2001 IEEE International Joint Conference on Neural Networks, Washington DC, vol. 3, pp. 1772–1777. IEEE, Piscataway (2001)

Orthogonalization and Thresholding Method for a Nonparametric Regression Problem

Katsuyuki Hagiwara

Faculty of Education, Mie University,
1577 Kurima-Machiya-cho, Tsu 514-8507 Japan
hagi@edu.mie-u.ac.jp

Abstract. In this article, we proposed training methods for improving the generalization capability of a learning machine that is defined by a weighted sum of many fixed basis functions and is used as a nonparametric regression method. In the basis of the proposed methods, vectors of basis function outputs are orthogonalized and coefficients of the orthogonal vectors are estimated instead of weights. The coefficients are set to zero if those are less than predetermined threshold levels which are theoretically reasonable under the assumption of Gaussian noise. We then obtain a resulting weight vector by transforming the thresholded coefficients. When we apply an eigen-decomposition based orthogonalization procedure, it yields shrinkage estimators of weights. If we employ the Gram-Schmidt orthogonalization scheme, it produces a sparse representation of a target function in terms of basis functions. A simple numerical experiment showed the validity of the proposed methods by comparing with other alternative methods including the leave-one-out cross validation.

1 Introduction

This article considers a regression method using a learning machine that is defined by a linear combination of fixed basis functions. Especially, we focus on a machine in which the number of basis functions, or equivalently, the number of adjustable weights is identical to the number of training data. This problem is viewed as a nonparametric regression method in statistics. In machine learning, such a formulation is employed in support vector machines(SVMs) in which the kernel trick with the representer theorem yields a linear combination of kernel functions; e.g. [2]. In recent works, it has also been proposed that variations of SVMs, such as least squared support vector machines(LS-SVMs)[6] and relevance vector machines(RVMs)[7]. Since we adopt a squared error loss function in this article, our formulation of training is similar to LS-SVM and RVM rather than SVM.

In a nonparametric regression method, we need devices for improving the generalization capability of a trained machine. Each of LS-SVM and RVM adopts a particular regularizer while both of them employ basically quadratic forms of a weight vector. In RVMs, regularization parameters are assigned individually to

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 187–194, 2009.

weights and those are iteratively updated based on the Bayesian log evidence. Basis functions are removed if the corresponding weights are small enough. This ensures a sparse representation of a target function in terms of basis functions. [3] developed a method called LROLS(locally regularized orthogonal least squares) which is a modification of RVM. In LROLS, vectors of basis function outputs are orthogonalized by a modified Gram-Schmidt procedure and coefficients of the orthogonalized vectors are estimated as in RVM. The resulting weight vector is obtained by transforming the estimated coefficients. [4] also introduced an orthogonalization procedure. [4] has applied an eigen-decomposition based orthogonalization procedure and proposed shrinkage methods based on thresholding of coefficients of orthogonalized vectors, in which threshold levels are introduced to decide whether to remove or keep the corresponding orthogonal vectors. By the benefit of orthogonalization, the proposed threshold levels are theoretically reasonable under the assumption of Gaussian noise. A model selection problem is thus solved in [4] while it is not in LROLS. Unfortunately, the methods in [4] do not give us a sparse representation of a target function in terms of basis functions. This article extends the idea in [4]. We derive a general form of a thresholding method on orthogonal coefficients and give the concrete implementations which include a shrinkage method and a method for obtaining a sparse representation.

In Section 2, we describe a learning machine and its training procedure, in which orthogonalization of vectors of basis function outputs and thresholding of coefficients of the orthogonal vectors are formulated. In Section 3, we derive threshold levels for coefficients. In Section 4, we give training methods which are concrete implementations of the procedure established in Section 2 and 3. In Section 5, the proposed methods are compared with other alternative methods throughout a simple numerical experiment. Section 6 is devoted to conclusions.

2 Formulations of Machine and Training Procedure

2.1 Learning Machine

Let $\{(\boldsymbol{x}_i, y_i) : \boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}, i = 1, \ldots, n\}$ be a set of input-output training data. We assume that output data are generated by $y_i = h(\boldsymbol{x}_i) + e_i,\ i = 1, \ldots, n$, where h is a fixed function on $\mathbb{R}^d$ and $e_1, \ldots, e_n$ are i.i.d. sequence of Gaussian noise having $N(0, \sigma^2)$; i.e. a Gaussian distribution with mean zero and variance σ^2. We refer to h by a target function or true function.

We consider a curve-fitting problem by using a machine whose output for $\boldsymbol{x} \in \mathbb{R}^d$ is given by

$$f_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{j=1}^{n} w_j g_j(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{w} = (w_1, \ldots, w_n) \in \mathbb{R}^n$ is a weight vector. (1) is a linear combination of fixed basis functions, in which the number of weights is identical to that of training data. $g_j(\boldsymbol{x})$ can be written by $g(\boldsymbol{x}, \boldsymbol{x}_j)$ if we employ kernel functions as basis functions.

Let G be an $n \times n$ matrix whose (i,j) element is $g_j(\boldsymbol{x}_i)$. We define $\boldsymbol{g}_j = (g_j(\boldsymbol{x}_1), \ldots, g_j(\boldsymbol{x}_n))'$ which is the jth column vector of G. $'$ denotes the transpose. We assume that $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_n$ are linearly independent. Also we redefine $\boldsymbol{w}$ as a vertical vector; i.e. $\boldsymbol{w} = (w_1, \ldots, w_n)'$. We define $\boldsymbol{f}_{\boldsymbol{w}} = (f_{\boldsymbol{w}}(\boldsymbol{x}_1), \ldots, f_{\boldsymbol{w}}(\boldsymbol{x}_1))$ which is written by $\boldsymbol{f}_{\boldsymbol{w}} = G\boldsymbol{w}$. We also define $\boldsymbol{y} = (y_1, \ldots, y_n)'$, $\boldsymbol{e} = (e_1, \ldots, e_n)'$ and $\boldsymbol{h} = (h(\boldsymbol{x}_1), \ldots, h(\boldsymbol{x}_n))'$.

2.2 Transformation of G

$\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_n\}$ can be regarded as a coordinate system since $\boldsymbol{f}_{\boldsymbol{w}}$ is represented by a weighted sum of those. In this article, we introduce another coordinate system which is obtained by a transformation of original coordinates and is an orthogonal system.

Let Q be an invertible $n \times n$ matrix. We define $A = GQ$ and denote the jth column vector of A by $\boldsymbol{a}_j$. We assume that $A'A$ is a diagonal matrix by an appropriate choice of Q. We define $\mathit{\Gamma} = A'A$, where $\mathit{\Gamma} = \mathrm{diag}(\gamma_1, \ldots, \gamma_n)$. $\mathrm{diag}(\gamma_1, \ldots, \gamma_n)$ denotes a diagonal matrix whose (j,j) element is γ_j. $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$ is thus a set of orthogonal vectors. Since $A'A = Q'(G'G)Q$, a transformation by Q implements a diagonalization of $G'G$ which is the Gram matrix of $\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_n\}$. For given input data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ and fixed basis functions $g_1, \ldots, g_n$, Q can be viewed as a matrix that transforms $\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_n\}$ into $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$. We call $\boldsymbol{a}_j$ an orthogonal component or orthogonal basis vector. We define

$$\boldsymbol{v} = (v_1, \ldots, v_n)' = Q^{-1}\boldsymbol{w}. \tag{2}$$

We can then write $\boldsymbol{f}_{\boldsymbol{w}} = G\boldsymbol{w} = A\boldsymbol{v}$. We call $\boldsymbol{v}$ an orthogonal coefficient vector.

2.3 Estimation of Orthogonal Coefficient Vector

In nonparametric regression methods, it is natural to introduce a regularization method to avoid over-fitting and stabilize a training process. In a regularization method, we obtain an estimator of $\boldsymbol{v}$ that minimizes a regularized cost function defined by $C(\boldsymbol{v}) = \|\boldsymbol{y} - A\boldsymbol{v}\|^2 + \boldsymbol{v}'\mathit{\Lambda}\boldsymbol{v}$, where the first term is an error function and the second term is a regularization term or regularizer. $\mathit{\Lambda}$ is an n-dimensional diagonal matrix defined by $\mathit{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, where $\lambda_j \geq 0$ for all j. Note that we may introduce a regularizer only for stabilizing a training procedure while the generalization capability is improved by another source. However, we need not introduce a regularizer and will actually apply another technique for the stabilization problem in below. In this case, we set $\lambda_j = 0$ for all j. A regularized estimator that minimizes $C(\boldsymbol{v})$ is defined by $\widehat{\boldsymbol{v}}$. By simple calculations, we have

$$\widehat{\boldsymbol{v}} = (\mathit{\Gamma} + \mathit{\Lambda})^{-1}A'\boldsymbol{y} \tag{3}$$

since $A'A = \mathit{\Gamma}$. By the definition of $\mathit{\Gamma}$ and $\mathit{\Lambda}$, we can write $\widehat{v}_j = \boldsymbol{a}_j'\boldsymbol{y}/(\gamma_j + \lambda_j)$ for $j = 1, \ldots, n$.

2.4 Thresholding of Orthogonal Coefficient Vector

By introducing a set of threshold levels defined by $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)'$, we consider to apply a hard thresholding operator $T_{\boldsymbol{\theta}}$ on an orthogonal coefficient vector. An orthogonal coefficient vector after thresholding is given by $T_{\boldsymbol{\theta}}(\boldsymbol{v}) = (T_{\theta_1}(v_1), \ldots, T_{\theta_n}(v_n))'$, where $T_{\theta_j}(v_j)$ is equal to v_j if $v_j^2 > \theta_j$ and is equal to 0 if $v_j^2 \le \theta_j$ for $j = 1, \ldots, n$. For the regularized estimator given by (3), we define $\overline{\boldsymbol{v}} = T_{\boldsymbol{\theta}}(\widehat{\boldsymbol{v}})$. By (2), we then obtain $\overline{\boldsymbol{w}} = Q\overline{\boldsymbol{v}}$ which is a resulting weight vector in our method.

3 Thresholding Levels

3.1 Statistical Property of $\widehat{\boldsymbol{v}}$

We denote a set of training inputs by $\boldsymbol{\xi} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. We assume that a target function is realizable in terms of $\{f_{\boldsymbol{w}} | \boldsymbol{w} \in \mathbb{R}^n\}$, i.e. there exists $\boldsymbol{w}^*$ such that $\boldsymbol{h} = G\boldsymbol{w}^* = A\boldsymbol{v}^*$, where $\boldsymbol{v}^* = (v_1^*, \ldots, v_n^*)' = Q\boldsymbol{w}^*$. Under the assumption of Gaussian noise, the conditional distribution of $\widehat{\boldsymbol{v}}$ given $\boldsymbol{\xi}$ is shown to be $\widehat{\boldsymbol{v}} \sim N(\boldsymbol{\nu}, S|\boldsymbol{\xi})$, where $\boldsymbol{\nu} = (\mu_1 v_1^*, \ldots, \mu_n v_n^*)'$ and $S = \sigma^2(\Gamma + \Lambda)^{-2}\Gamma = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_n^2)$ in which $\mu_j = \gamma_j/(\gamma_j + \lambda_j)$ and $\sigma_j^2 = \sigma^2\gamma_j/(\gamma_j + \lambda_j)^2$. Since S is diagonal, $\widehat{v}_j \sim N(\mu_j v_j^*, \sigma_j^2)$, $j = 1, \ldots, n$ hold and those are independent.

3.2 Thresholding Operation with Threshold Levels

We assume that a sparse representation of $\boldsymbol{h}$ in terms of orthogonal basis vectors which are column vectors of A. We define $V^* = \{j | v_j^* \ne 0,\ 1 \le j \le n\}$. We refer to $\{\boldsymbol{a}_j | j \in V^*\}$ by true components. Let K_n^* be the cardinality of V^*. The sparse representation implies that K_n^* is very small by comparing with n. We also define $V = \{j | v_j^* = 0,\ 1 \le j \le n\}$. Let K_n be the cardinality of V, where $K_n = n - K_n^*$. Orthogonal components with indexes in V do not relate to a target function and relate only to noise in training. We thus refer to $\{\boldsymbol{a}_j | j \in V\}$ by noise components.

We define $C_{n,\epsilon} = (2 + \epsilon)\log n$, where ϵ is a constant. Let δ be an arbitrary positive constant below. We consider a thresholding operator $T_{\boldsymbol{\theta}_{n,\delta}}$ in which $\boldsymbol{\theta}_{n,\delta} = (\theta_{1,n,\delta}, \ldots, \theta_{n,n,\delta})'$ and

$$\theta_{j,n,\delta} = \sigma_j^2 C_{n,\delta},\ j = 1, \ldots, n. \tag{4}$$

We state theoretical implications of these threshold levels without proofs while the validity of implications is easily confirmed from the extreme value theory[5].

(1) $\mathbb{P}\left[\bigcup_{j \in V} \left\{\widehat{v}_j^2 > \theta_{j,n,\delta}\right\}\right]$ goes to zero as $n \to \infty$.

(2) $\mathbb{P}\left[\bigcap_{j \in V} \left\{\widehat{v}_j^2 \le \theta_{j,n,-\delta}\right\}\right]$ goes to zero as $n \to \infty$ if $K_n \ge \rho n$ for $\rho \in (0, 1]$, or equivalently, $K_n^* \le (1 - \rho)n$.

The first fact tells us that, for any $j \in V$, $\widehat{v}_j^2$ cannot exceed $\theta_{j,n,\delta}$ with high probability when n is large. If we employ $\boldsymbol{\theta}_{n,\delta}$ as a set of component-wise threshold levels then the thresholding procedure surely remove noise components. On the other hand, the second fact tells us that there are some noise components for which $\widehat{v}_j^2 > \theta_{j,n,-\delta}$ with high probability when n is large. In other words, there exist some noise components for which $\widehat{v}_j^2$ are close to $\theta_{j,n,0}$. Thus, we can not distinguish a true component from a noise component if the corresponding $\widehat{v}_j^2$ is happen to be less than $\theta_{j,n,-\delta}$. Since δ can be taken as an arbitrary small value, $\theta_{j,n,0}$, $j = 1, \ldots, n$ are critical and reasonable levels for deciding whether to remove or keep orthogonal components if $K_n \geq \rho n$. $K_n \geq \rho n$ is fulfilled if we assume a sparse representation of a target function in terms of orthogonal components.

4 Implementations

4.1 Estimation of Noise Variance

Based on the previous discussion, we employ $\boldsymbol{\theta}_{n,0}$ as a vector of component-wise threshold levels. We thus apply $T_{\boldsymbol{\theta}_{n,0}}$ as a thresholding operator. In practical applications of $T_{\boldsymbol{\theta}_{n,0}}$, we need an estimate of noise variance in (4). Fortunately, in nonparametric regression methods, [1] suggested to apply $\widehat{\sigma}^2 = \boldsymbol{y}'(I_n - H)^2\boldsymbol{y}/\mathrm{trace}[(I_n - H)^2]$ as an estimate of σ^2, where $H = A(\mathit{\Gamma} + \mathit{\Lambda})^{-1}A'$ and I_n denotes the $n \times n$ identity matrix.

4.2 Orthogonalization Procedure

In this article, we consider two procedures of determining Q by which $A'A$ becomes a diagonal matrix, or equivalently, $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$ becomes a set of orthogonal vectors.

The first procedure is based on eigen-decomposition of $G'G$. We choose Q in which the kth column vector is the kth eigen vector of $G'G$. Q is then an orthonormal matrix; i.e. $Q^{-1} = Q'$. By this choice of Q, the column vectors of $A = GQ$, which are $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$, are orthogonal and actually $A'A = (GQ)'GQ = Q'(G'G)Q = \mathit{\Gamma}$, where γ_k is the kth eigen value of $G'G$. Since G is non-degenerate, we have $\gamma_k > 0$ for any k. Without loss of generality, we assume that $\gamma_1 \geq \gamma_2 \geq \cdots \geq \gamma_n > 0$. We refer to this procedure as ED.

The second procedure is based on the Gram-Schmidt orthogonalization for $G = (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_n)$, which is also employed in [3] for achieving a sparse representation also in terms of original basis vectors. In the Gram-Schmidt orthogonalization procedure, $\boldsymbol{g}_k$ is orthogonalized based on $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{k-1}$. We assume that $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{k-1}$ are orthogonal vectors which are already obtained by the Gram-Schmidt orthogonalization of $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{k-1}$. At the kth step, we define $p_{j,k} = \langle \boldsymbol{q}_j, \boldsymbol{g}_k \rangle / \|\boldsymbol{q}_j\|^2$, $j = 1, \ldots, k-1$, where $\langle \cdot, \cdot \rangle$ denotes the Euclidean inner product. We then obtain $\boldsymbol{q}_k = \boldsymbol{g}_k - \sum_{j=1}^{k-1} p_{j,k}\boldsymbol{q}_j$ which is orthogonal to $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{k-1}$. This procedure is successively applied by starting from $\boldsymbol{q}_1 = \boldsymbol{g}_1$.

We define $P_1 = I_n$. Let P_k, $k = 2, \ldots, n$ be an $n \times n$ matrix whose (j, k) element is $p_{j,k}$ if $j = 1, \ldots, k-1$, is 1 if $j = k$ and is 0 otherwise. We define $Q_k = \prod_{j=1}^{k} P_j$. We then have $GQ_k = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k, \boldsymbol{g}_{k+1}, \cdots, \boldsymbol{g}_n)$; i.e. multiplication of Q_k achieves the Gram-Schmidt orthogonalization up to the kth step. We choose Q_n as Q. Then $A = GQ = (\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n)$. We thus set $\boldsymbol{a}_k = \boldsymbol{q}_k$ and $\gamma_k = \|\boldsymbol{q}_k\|^2$. We refer to this procedure as GS. In each step of the orthogonalization in [3], a basis function is selected to reduce the residual error maximally, by which it is difficult to construct a criterion for choosing an set of effective basis functions. Note that our threshold levels cannot be applied to this procedure. In our GS, at the kth step, we calculate $p_{j,i}$, $\boldsymbol{q}_i$ and γ_i for all i in $\{k, \ldots, n\}$. We then find $c_k = \arg\max_{k \le i \le n} \gamma_i$. We set $p_{j,k} = p_{j,c_k}$ and $\boldsymbol{q}_k = \boldsymbol{q}_{c_k}$, hence $\boldsymbol{g}_k = \boldsymbol{g}_{c_k}$. This process chooses the kth orthogonal component which is possibly disrelated to the previously chosen components. Note that the proposed threshold levels are valid in this case since we do not utilize training outputs for choosing an orthogonal component at each step.

We refer to hard thresholding methods using the ED and GS procedures by HTED and HTGS respectively. In HTED and HTGS, we define J as an index such that $\overline{v}_j = 0$ for all $J < j \le n$. HTED and HTGS yield different types of estimates. The trained weight vector of HTED is shown to be a shrinkage estimator due to the orthonormality of Q and the definition of our hard thresholding method. In the GS procedure, the number of non-zero weights is also J by its definition. In other words, HTGS works as a thresholding method also on an original representation when $J < n$. HTGS is thus possible to produce a sparse representation if J is small.

4.3 Modifications of HTED and HTGS

In this article, we further consider modifications of HTED and HTGS. Orthogonal components with large γ_j's mainly contribute for machine outputs. If a hard thresholding method happen to remove such components, it is possible to yield a large bias. It is thus safety to avoid the unexpected remove and keep such components. We then consider to keep all of J orthogonal components and set $\overline{\boldsymbol{v}}_J = (\widehat{v}_1, \ldots, \widehat{v}_J)'$. In other words, this procedure is a stopping point search, which is J here, in increasing order of magnitudes of γ_j's. HTED and HTGS with this modification are referred to as HTED2 and HTGS2 respectively.

4.4 Numerical Stability Problem

Although a set of basis functions is assumed to be linearly independent, it may arise a problem of numerical instability of trained weights. In our methods, it is caused by estimation of coefficients for orthogonal components with small values for γ_j's. To avoid the numerical instability, one can take small values for regularization parameters. Here, we set $\lambda_j = 0$ for all j and rather consider to use a set of stable orthogonal components for which $\gamma_j > \eta$, where η is a fixed positive constant. This reduces a computational cost for estimating coefficients, thresholding and inverse transform for obtaining a weight vector. We choose a small value for η to just ensure the numerical stability. The number of candidates of stable components is then relatively large. However, in those candidates, effective components are automatically selected by thresholding.

5 Numerical Experiments

We compare performances of the proposed methods to those of the leave-one-out cross validation(LOOCV), RVM and LROLS.

A target function is $h(x) = 3.0\,\mathrm{sinc}(8.0\,x)$ for $x \in \mathbb{R}$. $x_1, \ldots, x_n$ are randomly drawn in the interval $[-1, 1]$. We set $\sigma^2 = 1$. Basis functions are Gaussian basis functions that are defined by $g_j(x) = \exp(-(x - x_j)^2/(2\tau^2)),\ j = 1, \ldots, n$. We trained a machine for n training samples, in which a value of τ is chosen from $\{0.02, 0.05, 0.1, 0.2, 0.5\}$ by the 10-folds cross validation. After training, test error is measured by the mean squared error between outputs of a target function and a trained machine on 2000 equally spaced input points in $[-1, 1]$. We repeat this procedure for 100 sets of training samples and then obtain the averaged test error.

In LOOCV, we assume that a squared error loss and an ℓ_2-regularizer with a single regularization parameter. A set of candidate values for a regularization parameter in LOOCV is $\{k \times 10^j;\ k = 1, 2, 5,\ j = -4, -3, \ldots, 2\}$. Parameters of RVM are λ_R, K_R and η_R which are an initial common value for regularization parameters, the number of repetitions for updating regularization parameters and a threshold level for a measure of certainty of weights respectively[7]. We set $\lambda_R = 10^{-3}$, $K_R = 100$ and $\eta_R = 10^{-12}$. Parameters of LROLS are λ_L, K_L and η_L which are an initial common value for regularization parameters, the number of repetitions for updating regularization parameters and a threshold level for the norm of orthogonal components. We set $\lambda_L = 10^{-3}$, $K_L = 20$ and $\eta_L = 10^{-12}$. In RVM and LROLS, η_R and η_L are criteria for removing basis functions. For our methods, we set $\lambda_j = 0$ for $j = 1, \ldots, n$ and $\eta = 10^{-12}$.

In Table 1, we show the averaged test errors and the numbers of remaining basis functions when $n = 200$ and 400. In the table, we can see that HTED2 is superior to the other methods in terms of the averaged test error while the

Table 1. Averaged test errors and the numbers of remaining basis functions

	test errors		the number of basis functions	
n	200	400	200	400
LOOCV	0.0617±0.0243	0.0347±0.0210	200	400
HTED	0.0552±0.0157	0.0302±0.0094	200	400
HTED2	0.0482±0.0112	0.0269±0.0082	200	400
HTGS	0.0869±0.0240	0.0435±0.0137	6.22±0.71	6.78±0.74
HTGS2	0.0647±0.0175	0.0303±0.0095	6.27±0.68	6.69±0.79
RVM	0.0653±0.0219	0.0356±0.0165	48.24±7.91	89.24± 14.99
LROLS	0.0648±0.0185	0.0307±0.0085	17.15±3.97	17.92±3.96

differences of the averaged test errors are almost within the standard deviations. We can also see that HTGS and HTGS2 are superior to the other methods in terms of sparseness. However, HTGS is worse than HTGS2 in terms of the averaged test errors. This gap is due to a bias caused by unexpected removes of contributed components. For RVM and LROLS, there may be better choices of the parameter values while there are no systematic choices of them. Especially, determining the number of basis functions, which is controlled by η_R in RVM and η_L in LROLS, is still an important problem in RVM and LROLS while some heuristics may be considered. HTGS and HTGS2 solve this problem based on the benefit of orthogonalization.

6 Conclusions

In this article, we proposed shrinkage and thresholding methods for a nonparametric regression problem. The proposed methods are based on orthogonalization of vectors of basis function outputs. By introducing a hard thresholding method on coefficients of orthogonal components, we derived HTED and HTED2 as shrinkage methods and HTGS and HTGS2 as thresholding methods on original representation. The important point is that the proposed threshold levels in thresholding are theoretically reasonable under the assumption of Gaussian noise and existence of a sparse representation of a target function in terms of orthogonal components. Throughout a simple numerical experiment, performances of the proposed methods are compared to those of LOOCV, RVM and LROLS. As a result, we found that HTED2 is superior or comparable to the other methods in terms of the generalization capability and HTGS2 is preferable to the other methods in terms of sparseness, in which sparseness may unfortunately not imply a better generalization performance.

References

1. Carter, C.K., Eagleson, G.K.: A comparison of variance estimators in nonparametric regression. J.R.Statist.Soc. B 54, 773–780 (1992)
2. Cristianini, N., Shawe-Taylor, J.: An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, Cambridge (2000)
3. Chen, S.: Local regularization assisted orthogonal least squares regression. Neurocomputing 69, 559–585 (2006)
4. Hagiwara, K.: An orthogonalized thresholding method for a nonparametric regression under Gaussian noise. In: Ishikawa, M., Doya, K., Miyamoto, H., Yamakawa, T. (eds.) ICONIP 2007, Part I. LNCS, vol. 4984, pp. 537–546. Springer, Heidelberg (2008)
5. Leadbetter, M.R., Lindgren, G., Rootz'en, H.: Extremes, and related properties of random sequences and processes. Springer, Heidelberg (1983)
6. Suykens, J.A.K., Brabanter, J.D., Lukas, L., Vandewalle, J.: Weighted least squares support vector machines: robustness and sparse approximation. Neurocomputing 48, 85–105 (2002)
7. Tipping, M.E.: Sparse Bayesian learning and the Relevance vector machine. Journal of Machine Learning Research 1, 211–244 (2001)

Analysis of Ising Spin Neural Network with Time-Dependent Mexican-Hat-Type Interaction

Kazuyuki Hara[1], Seiji Miyoshi[2], Tatsuya Uezu[3], and Masato Okada[4,5]

[1] Tokyo Metropolitan College of Industrial Technology, 1-10-40, Higashi-oi,
Shinagawa, Tokyo, 140-0011, Japan
hara@s.metro-cit.ac.jp

[2] Faculty of Engineering Science, Kansai University,
3-3-35, Yamate-cho, Suita, Osaka, 564-8680, Japan
miyoshi@ipcku.kansai-u.ac.jp

[3] Graduate School of Humanities and Sciences, Nara Women's University,
Kitauoya Nishicho, Nara-shi, Nara, 630-8506, Japan
uezu@cc.nara-wu.ac.jp

[4] Graduate School of Frontier Sciences, The University of Tokyo,
5-1-5, Kashiwanoha, Kashiwa-shi, Chiba, 277-8561, Japan
okada@k.u-tokyo.ac.jp

[5] Brain Science Institute, Riken, 2-1, Hirosawa, Wako, Saitama, 351-0198, Japan

Abstract. We analyzed the equilibrium states of an Ising spin neural network model in which both spins and interactions evolve simultaneously over time. The interactions are Mexican-hat-type, which are used for lateral inhibition models. The model shows a bump activity, which is the locally activated network state. The time-dependent interactions are driven by Langevin noise and Hebbian learning. The analysis results reveal that Hebbian learning expands the bistable regions of the ferromagnetic and local excitation phases.

1 Introduction

Neural networks that can model local excitation are often described by lateral inhibition type networks. The lateral inhibition represents recurrent excitation with nearby neurons and inhibition between distant neurons: it is also referred to as "Mexican-hat-type interaction".

Hamaguchi, Hatchett, and Okada built a statistical mechanics of a neural network on a one-dimensional ring with disordered lateral inhibition and analyzed its equilibrium state[1]. Keeping the difficulty of analyzing the replica-symmetry breaking (RSB) using the signal to noise (SN) analysis in mind, they defined the Hamiltonian of their system and discussed the effects of disordered lateral inhibition by using the replica method. They showed that there are four phases: paramagnetic (P), ferromagnetic (F), local excitation (L), and spin-glass (SG). They also showed that there is a bump activity in their model, which is the locally activated network state. It is stable depending on the input and configuration of the interactions of the network[1].

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 195–202, 2009.

Recurrent neural networks with time-dependent interaction have been investigated by several groups[2],[3],[4]. Coolen et al. shows that both neurons and synaptic interactions (or coupling) evolve simultaneously over time in accordance with specific coupled stochastic equations. They assumed that the time-scale of the neuron dynamics is sufficiently shorter than that of the interactions. When neurons are updated, the interactions are assumed to be fixed to the instantaneous values in the equilibrium state. The coupling dynamics are described by the Langevin equations including an Hebbian learning term and a Langevin noise term. Coolen et al. demonstrated that the replica method can be used to calculate the equilibrium state of the network through time-dependent interactions. Their results showed that the phase transition temperature is changed by the time-dependent interactions. Dotsenko et al. introducing time-dependent interaction increases the capacity of the network.

We analyzed the equilibrium states of an Ising spin neural network with time-dependent Mexican-hat-type interaction. The concept of capacity is not defined in the network. A bump activity in such a locally activated network state and is stable depending on the input and configuration of the interactions of the network. We investigated how the stability of a bump activity is changed by introducing the time-dependent interaction. We also investigated the effect of Hebbian learning in the coupling dynamics on the equilibrium states of the network.

2 Model Definitions

The Ising spin neural network we used is modeled by an N-neuron state vector $\boldsymbol{\sigma} = (\sigma_{\theta_1}, \sigma_{\theta_2}, \ldots, \sigma_{\theta_N}) \in \{-1, 1\}^N$. Here $\sigma_{\theta_i} = 1$ if neuron i fires, and $\sigma_{\theta_i} = -1$ if it is at rest. Neuron i is located at angle $\theta_i = \frac{2\pi i}{N} - \pi$ on a one-dimensional ring indexed by $\theta_i \in [\pi, \pi)$. The Hamiltonian of the system studied is

$$H(\boldsymbol{\sigma}) = -\frac{1}{2} \sum_{\theta_i \neq \theta_j} J_{\theta_i\theta_j} \sigma_{\theta_i} \sigma_{\theta_j} - h \sum_{\theta_i} \sigma_{\theta_i}. \tag{1}$$

where h is a common external input to the neurons. We assume that the coupling dynamics slow compared to the spin dynamics. A Langevin equation is used for the coupling dynamics.

$$\tau \frac{dJ_{\theta_i\theta_j}}{dt} = \frac{1}{N}[\epsilon < \sigma_{\theta_i}\sigma_{\theta_j} >_{sp} + K_{\theta_i\theta_j}] - \mu J_{\theta_i\theta_j} + \sqrt{\frac{\tau}{N}} \eta_{\theta_i\theta_j}(t) \tag{2}$$

with $\tau << 1$. In the adiabatic limit $\tau \to \infty$, the term $< \sigma_i \sigma_j >_{sp}$, representing spin correlations associated with coupling J_{ij}, is the average over the Boltzmann distribution of the spins, given instantaneous couplings $\boldsymbol{J}$. The ϵ represents the strength of Hebbian learning. This model is equivalent to that of Hamaguchi when $\epsilon = 0$[1]. The $\eta_{\theta_i\theta_j}(t)$ represents white Gaussian noise contribution of zero mean and covariance $< \eta_{\theta_i\theta_j}(t)\eta_{\theta_k\theta_l}(t') >= 2\tilde{T}\delta_{\theta_i\theta_k}\delta_{\theta_j\theta_l}\delta(t-t'), i < j, k < l$ with associate $\tilde{T} = \tilde{\beta}^{-1}$. Interaction $J_{\theta_i\theta_j}$ is defined to be lateral inhibition, and it is a function only of $(\theta_i - \theta_j)$: $K_{\theta_i\theta_j} = K_0 + K_1 \cos(\theta_i - \theta_j)$ where K_0 is a uniform ferromagnetic interaction, and K_1 is a lateral inhibition type interaction. This model can be considered to be a Fourier series expansion of the interaction function with a lower order and periodic boundary condition.[1] The time-scale of coupling dynamics is very long compared to the spin dynamics, so couplings can be considered to be quenched state in the spin dynamics. We assume that K_0 and K_1 are non-negative real numbers.

3 Replica Calculation of the Free Energy and Order Parameters

The free energy per neuron and relevant order parameters are calculated using the replica method. The effective Hamiltonian $\mathcal{H}$ is represented as

$$\mathcal{H} = -\frac{\epsilon}{\beta} \ln Z_\beta - \sum_{\theta_i<\theta_j} K_{\theta_i\theta_j} J_{\theta_i\theta_j} + \frac{N\mu}{2} J^2_{\theta_i\theta_j}, \tag{3}$$

so eq. (2) can be rewritten as

$$N\tau \frac{dJ_{\theta_i\theta_j}}{dt} = -\frac{\partial \mathcal{H}}{\partial J_{\theta_i\theta_j}} + \sqrt{N\tau}\eta_{\theta_i\theta_j}(t). \tag{4}$$

The equilibrium distribution of $\{J_{\theta_i\theta_j}\}$, $P_{eq}(\{J_{\theta_i\theta_j}\})$ is a Boltzmann distribution and is specified by the partition function $\tilde{Z}_{\tilde{\beta}}$ and free energy $\tilde{F}_{\tilde{\beta}}$. The partition function related to the effective Hamiltonian $\mathcal{H}$ is given by

$$\begin{aligned}\tilde{Z}_{\tilde{\beta}} &= \int d\boldsymbol{J} \exp(-\tilde{\beta}\mathcal{H}) \\ &= \int \left[\prod_{\theta_i<\theta_j} \sqrt{\frac{N\mu\tilde{\beta}}{2\pi}} dJ_{\theta_i\theta_j}\right] Z_\beta^{\epsilon\frac{\tilde{\beta}}{\beta}} \\ &\times \exp\left(\tilde{\beta} \sum_{\theta_i<\theta_j} K_{\theta_i\theta_j} J_{\theta_i\theta_j} - \frac{\tilde{\beta}N\mu}{2} \sum_{\theta_i<\theta_j} J^2_{\theta_i\theta_j}\right).\end{aligned} \tag{5}$$

The Z_β is the conventional partition function with inverse temperature $\beta = T^{-1}$ related to the spins.

$$Z_\beta = \exp(-\beta H) = \exp\left(\beta \sum_{\theta_i<\theta_j} J_{\theta_i\theta_j} \sigma_{\theta_i}\sigma_{\theta_j} + \beta h \sum_{\theta_i} \sigma_{\theta_i}\right). \tag{6}$$

As shown by eq.(5), the spin dynamics and coupling dynamics are controlled by the individual inverse temperatures of β and $\tilde{\beta}$, respectively. We rewrite $\epsilon\frac{\tilde{\beta}}{\beta} = n$, so eq. (5) has the same form as the Sherrington-Kirkpatrick (SK) model, so we can calculate the free energy per neuron and relevant order parameters of the system through the replica method. By introducing replicas $\boldsymbol{\sigma}^1, \boldsymbol{\sigma}^2, \cdots \boldsymbol{\sigma}^n$, we can express Z_β^n as

$$Z_\beta^n = \mathrm{Tr}_{\boldsymbol{\sigma}} \exp\left(-\beta \sum_{\alpha=1}^{n} H(\boldsymbol{\sigma}^\alpha)\right), \tag{7}$$

were Tr means the sum of all possible combinations of $\boldsymbol{\sigma}$, and α is the replica index. The partition function is then given by

$$\begin{aligned}\tilde{Z}_{\tilde{\beta}} = \mathrm{Tr}_{\boldsymbol{\sigma}} \int \prod_{\alpha<\beta} dq^{\alpha\beta} \prod_{\alpha} dM_0^\alpha \prod_{\alpha} dM_C^\alpha \prod_{\alpha} dM_S^\alpha \\ \times \exp\left\{-\frac{N\tilde{\beta}^2 J^2}{2} \sum_{\alpha<\beta} (q^{\alpha\beta})^2 - \frac{N\tilde{\beta}K_0\mu}{2} \sum_{\alpha} (M_0^\alpha)^2 \right. \\ \left. - \frac{N\tilde{\beta}K_1\mu}{2} \sum_{\alpha} ((M_C^\alpha)^2 + (M_S^\alpha)^2) + L + \frac{N\tilde{\beta}^2 J^2 n}{4}\right\}. \end{aligned} \tag{8}$$

where

$$\begin{aligned} L = \sum_{\theta_i} \left\{ J^2\beta^2 \sum_{\alpha<\beta} \sigma_{\theta_i}^\alpha \sigma_{\theta_i}^\beta q^{\alpha\beta} + \beta K_0 \sum_{\alpha} M_0^\alpha \sigma_{\theta_i} \right. \\ \left. + \beta K_1 \sum_{\alpha} (M_C^\alpha \cos\theta_i + M_S^\alpha \sin\theta_i)\, \sigma_{\theta_i}^\alpha + \beta h \sum_{\alpha} \sigma_{\theta_i}^\alpha \right\}, \end{aligned} \tag{9}$$

and $J = \sqrt{1/\tilde{\beta}\mu}$. We define four order parameters.

$$\begin{aligned} q = N^{-1} \sum_{\theta_i} \sigma_{\theta_i}^\alpha \sigma_i^\beta,\ M_0 = N^{-1} \sum_{\theta_i} \sigma_{\theta_i}, \\ M_C = N^{-1} \sum_{\theta_i} \sigma_{\theta_i} \cos\theta_i,\ M_S = N^{-1} \sum_{\theta_i} \sigma_{\theta_i} \sin\theta_i \end{aligned}$$

where M_0 is the magnetization parameter, q is the spin-glass order parameter, and M_C and M_S are order parameters that show how spins are aligned in the same direction locally, but not globally. As Hamaguchi pointed out, the network state is in the bump state, or locally activated, if M_C or M_S is nonzero. We note that M_C and M_S depend on the position of the bump, but our interests are rather the size and the position of the bump. We thus introduce the following transformation to separate the size and position of the bump.

$$M_1^\alpha = \sqrt{(M_C^\alpha)^2 + (M_S^\alpha)^2},\ \phi^\alpha = \tan^{-1}(M_S^\alpha / M_C^\alpha).$$

The alternative order parameters, M_1 and ϕ are global measure of the activity profiles and indicate the degree of activity localization and its angle, respectively. We can now rewrite the third term of eq. (9) as $\beta K_1 \sum_\alpha M_1^\alpha [\cos(\theta_i - \phi^\alpha)] \sigma_{\theta_i}^\alpha$.

We assume replica symmetry, that is $q^{\alpha\beta} = q$,$M_0^\alpha = M_0$, $M_1^\alpha = M_1$, and $\phi^\alpha = \phi$. The introduction of the replica symmetry and Gaussian integral identity

$\exp(\frac{b^2}{2}) \equiv \int Dz_{\theta_i} \exp(bz_{\theta_i})$, $Dz_{\theta_i} = \frac{dz_{\theta_i}}{\sqrt{2\pi}} \exp(-\frac{z_{\theta_i}^2}{2})$ simplify $\mathrm{Tr}_{\boldsymbol{\sigma}} \exp(L)$ in Eq. (8) to

$$\begin{aligned}\mathrm{Tr}_{\boldsymbol{\sigma}} \exp(L) = \mathrm{Tr}_{\boldsymbol{\sigma}} \prod_{\theta_i} \int & Dz_{\theta_i} \exp\Big\{[\beta J\sqrt{q} z_{\theta_i} + \beta K_0 M_0 \\ & + \beta K_1 M_1 \cos(\theta_i - \phi) + \beta h] \sum_{\alpha} \sigma_{\theta_i}^{\alpha}\Big\} \\ & \times \exp\left(-\frac{Nn\beta^2 J^2}{2} q + \frac{Nn\beta^2 J^2}{4}\right) \\ & = \prod_{\theta_i} \int Dz_{\theta_i} [2\cosh \beta \tilde{H}(Z_{\theta_i}, \theta_i - \phi)]^n \\ & \times \exp\left(-\frac{Nn\beta^2 J^2}{2} q + \frac{Nn\beta^2 J^2}{4}\right),\end{aligned} \tag{10}$$

where $\tilde{H}(z_{\theta_i}, \theta_i) = J\sqrt{q} z_{\theta_i} + K_0 M_0 + K_1 M_1 \cos(\theta_i) + h$. From this, we get

$$\begin{aligned}\tilde{Z}_{\tilde{\beta}} = \int & dq dM_0 dM_c dM_s \exp\Big\{\frac{NnJ^2\beta^2}{4}[(1-n)q^2 - 2q + 1] \\ & - \frac{Nn\beta K_0}{2} M_0^2 - \frac{Nn\beta K_1}{2} M_1^2 \\ & + \sum_{\theta_i} \ln\left(\int Dz_{\theta_i} [2\cosh \beta \tilde{H}(z_{\theta_i}, \theta_i - \phi)]^n\right)\Big\}\end{aligned} \tag{11}$$

The summation of θ_i is now replaced by an integral over θ. We define the free energy per neuron by $\tilde{f}_{\tilde{\beta}} = -\frac{1}{\tilde{\beta} N} \ln \tilde{Z}_{\tilde{\beta}}$. We get the next equation by evaluating the integral at the saddle point.

$$\begin{aligned}-\tilde{\beta}\tilde{f}_{\tilde{\beta}} = & -\frac{\beta K_0 \mu n}{2} M_0^2 - \frac{\beta K_1 \mu n}{2} M_1^2 + \frac{\beta^2 J^2}{4} n(1 - 2q + (1-n)q^2) \\ & + \frac{1}{2\pi} \int_{-\pi}^{\pi} Dz \ln[2\cosh^n \beta \tilde{H}(z, \theta - \phi)].\end{aligned} \tag{12}$$

The order parameters are given by the saddle point equations of free energy $\tilde{f}_{\tilde{\beta}}$ through the saddle point method. We introduce variables $M_0\mu = m_0$, $M_C\mu = m_c$, $M_S\mu = m_S$, and $m_1 = \sqrt{m_C^2 + m_S^2}$, and then the order parameters are

$$m_0 = \int_{-\pi}^{\pi} \frac{d\theta}{2\pi} \frac{\int Dz \cosh^n \beta(\tilde{H}(z_\theta, \theta - \phi)) \tanh \beta(\tilde{H}(z_\theta, \theta - \phi))}{\int Dz \cosh^n \beta(\tilde{H}(z_\theta, \theta - \phi))}, \tag{13}$$

$$m_1 = \int_{-\pi}^{\pi} \frac{d\theta}{2\pi} \frac{\int Dz \cos(\theta - \varphi) \cosh^n \beta(\tilde{H}(z_\theta, \theta - \phi)) \tanh \beta(\tilde{H}(z_\theta, \theta - \phi))}{\int Dz \cosh^n \beta(\tilde{H}(z_\theta, \theta - \phi))}, \tag{14}$$

$$q = \int_{-\pi}^{\pi} \frac{d\theta}{2\pi} \frac{\int Dz \cosh^n \beta(\tilde{H}(z_\theta, \theta - \phi)) \tanh^2 \beta(\tilde{H}(z_\theta, \theta - \phi))}{\int Dz \cosh^n \beta(\tilde{H}(z_\theta, \theta - \phi))}, \tag{15}$$

where $\tilde{H}(z_\theta, \theta-\phi) = J\sqrt{q}z_\theta + \frac{K_0}{\mu}m_0 + \frac{K_1}{\mu}m_1\cos(\theta-\phi)$, $n = \epsilon\tilde{\beta}/\beta$, $J = 1/\sqrt{\tilde{\beta}\mu}$, $Dz \equiv \frac{dz}{\sqrt{2\pi}}\exp(-\frac{z^2}{2})$. These saddle point equations are similar to those of a neural network with disordered lateral inhibition[1].

4 Results

4.1 Temperature Dependence of Order Parameters

Figure 1 shows numerical solutions of the saddle point equations (Eqs. (13), (14), and (15)). We defined βK_0 and βK_1 as the inverse temperatures of spins, and we set β to 1. We set the inverse temperature of the Langevin noise, $\tilde{\beta}$ to 10, meaning that the Langevin noise was assumed to be small. We scanned the inverse temperature (K_1) from higher (right side of figure) to lower (left side). In the figure, the horizontal axis shows the inverse temperature K_1, and the vertical axis shows the order parameter m_1. We set $K_0 = 0$ and $\mu = 1$ in eq. (2). The strength of Hebbian learning is $\epsilon = 1.0$ is referred as "learning", and it is $\epsilon = 0.0$ is referred as "not-learning". The solid lines represent a stable solution, and dotted line represents an unstable solution. A stable solution is given by using the iterative method, and an unstable solution is given by the Newtonian method.

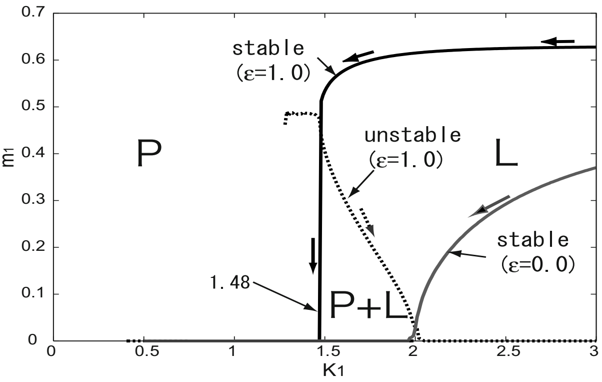


Fig. 1. Temperature dependence and phase transition of order parameter m_1

The figure shows that for "not-learning", the phase transition from Local phase ($m_1 \neq 0, q \neq 0, m_0 = 0$), refereed to as "L", to paramagnetic phase($m_0 = m_1 = q = 0$), referred to as "P", was a second order phase transition. However, for "learning", the transition was a first order one. The phase transition temperature from L to P for "not-learning" was $K_1 = 2.0$; it shifted to $K_1 = 1.48$ for "learning". Moreover, only a stable solution was obtained for "not-learning", while both unstable and stable solutions were obtained for "learning". The unstable solutions were obtained between $K_1 = 1.48$ and 2.0, and a stable solution

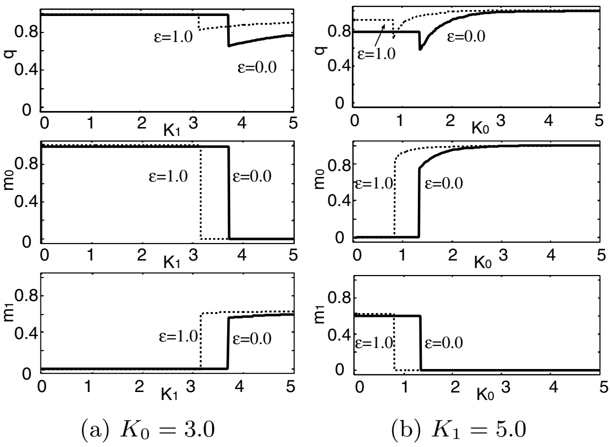


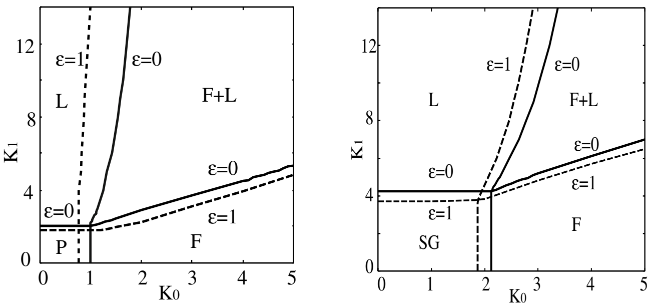
Fig. 2. Temperature dependence of order parameters q, m_0, and m_1

of $m_1 = 0$ was obtained in this interval. Bistable regions, referred to as “P+L”, were thus obtained. Similar phenomena were obtained for order parameter m_0.

Figure 2(a) shows the temperature dependence of the order parameters(q, m_0 and m_1 from the top) when the inverse temperature K_0, was set to 3, and Fig. 2(b) shows the dependence when K_1 was set to 5 (with K_0 scanned from 5 to 0). The inverse temperature of the Langevin noise was set to 10 in both cases. The $\epsilon = 0$ corresponds to “not-learning”, and $\epsilon = 1$ corresponds to results of “learning”. From these figures, we can see that (1) the phase transitions of $m_0 \neq 0 \to m_0 = 0$(Fig.2(a) middle), $m_0 = 0 \to m_0 \neq 0$(Fig.2(b) middle), $m_1 = 0 \to m_1 \neq 0$(Fig. 2(a) bottom), and $m_1 \neq 0 \to m_1 = 0$ (Fig.2(b) bottom) were the first order transitions for any strength of Hebbian learning ϵ, and (2) all the phase transition temperatures tended to be shifted to lower inverse temperatures. Figure 2(a) shows that the state of the system is L$(m_1 \neq 0, q \neq 0, m_0 = 0)$ for K_1=5, and Fig. 2(b) shows that the state is F$(m_0 \neq 0, q \neq 0, m_1 = 0)$ for $K_0 = 3$. This means that, at the point $(K_0, K_1) = (3, 5)$, both L and F are stable; that is, there is a bistable region.

4.2 Phase Diagram and Order Parameters

Figure 3(a) shows the phase diagram of the system when the inverse temperature of the Langevin noise was 10. Figure 3(b) shows diagram when it was 0.25. In these figures, P represents the paramagnetic phase($m_0 = m_1 = q = 0$), and SG represents the spin-glass phase($m_0 = m_1 = 0, q \neq 0$). Depending on the relative strength of K_0 and K_1, F or L are stable once they exceed a certain threshold. Between them, there are bistable regions, which are represented as F+L, where both the F and L phases are locally stable. The bistable regions shrink as $\tilde{\beta}$ decreases. Hamaguchi showed that when the coupling noises are relatively large, an SG phase appears instead of a P phase[1]. As shown by Fig. 3, the SG phase appears in the present model. When $\epsilon = 1$, the bistable region gradually widened



(a) $\tilde{\beta} = 10$

(b) $\tilde{\beta} = 0.25$

Fig. 3. Phase diagram with fixed $\tilde{\beta}$ in the intersection of $\{K_0, K_1\}$ plane: (a) is set to $\tilde{\beta} = 10$, and (b) is set to $\tilde{\beta} = 0.25$, respectively

compared to that with $\epsilon = 0$. This means that the bistable region becomes more stable in wider regions of the $\{K_0, K_1\}$ plane when Hebbian learning is used in the coupling dynamics.

5 Conclusions

We analyzed the equilibrium states of an Ising spin neural network with time-dependent Mexican-hat-type interaction using the replica method. We derived the saddle point equations for order parameters q, m_0 and m_1, and showed that they are similar to those of a neural network with disordered lateral inhibition[1]. We analyzed the phase transition of ferromagnetic interaction or lateral inhibition interaction. By solving the saddle point equations numerically, we showed that, with Hebbian learning, the phase transition is a first order transition, and that, without Hebbian learning, it is a second order phase transition. We then drew the phase diagram of the system for small and large Langevin noise. We found that there was a paramagnetic phase, a ferromagnetic phase, a local excitation phase, and a spin-glass phase. The paramagnetic phase appears when the Langevin noise is small, and the spin-glass phase appears when the noise is large. The bistable regions of the ferromagnetic and local excitation phases tended to widen with Hebbian learning in the coupled dynamics.

References

1. Hamaguchi, K., Hatchett, J.P.L., Okada, M.: Physical Review E 73, 051104 (2006)
2. Penny, R.W., Coolen, A.C.C., Sherrington, D.: J. Phys. A: Math. Gen. 26, 3681–3695 (1993)
3. Uezu, T., Coolen, A.C.C.: J. Phys. A: Math. Gen. 35, 2761–2809 (2001)
4. Dotsenko, V., Franz, S., Mézard, J.: Phys. A: Math. Gen. 27, 2351–2365 (1994)

Divided Chaotic Associative Memory for Successive Learning

Takahiro Hada and Yuko Osana

School of Computer Science
Tokyo University of Technology,
1401-1 Katakura, Hachioji, Tokyo, Japan
takahiro@osn.cs.teu.ac.jp, osana@cc.teu.ac.jp

Abstract. In this paper, we propose a Divided Chaotic Associative Memory for Successive Learning (DCAMSL). The proposed model is based on the Improved Chaotic Associative Memory for Successive Learning (ICAMSL) and the Divided Chaotic Associative Memory for Successive Learning using Internal Patterns (DCAMSL-IP) which were proposed in order to improve the storage capacity. In most of the conventional neural network models, the learning process and the recall process are divided, and therefore they need all information to learning in advance. However, in the real world, it is very difficult to get all information to learn in advance. So we need the model whose learning and recall processes are not divided. As such model, although some models have been proposed, their storage capacity is small. In the proposed DCAMSL, the learning process and the recall process are not divided and its storage capacity is larger than that of the conventional ICAMSL.

1 Introduction

Recently, neural networks are drawing much attention as a method to realize flexible information processing. Neural networks consider neuron groups of the brain in the creature, and imitate these neurons technologically. Neural networks have some features, especially one of the important features is that the networks can learn to acquire the ability of information processing.

In the filed of neural network, many models have been proposed such as the Hopfield network[1] and the Bidirectional Associative Memory[2]. In these models, the learning process and the recall process are divided, and therefore they need all information to learn in advance.

However, in the real world, it is very difficult to get all information to learn in advance. So we need the model whose learning and recall processes are not divided. As such model, Grossberg and Carpenter proposed the Adaptive Resonance Theory (ART)[3]. However, the ART is based on the local representation, and therefore it is not robust for damaged neurons. While in the field of associative memories, some models have been proposed[4] [5]. Since these models are based on the distributed representation, they have the robustness for damaged

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 203–211, 2009.

neurons. However, their storage capacity is very small because their learning processes are based on the Hebbian learning. In contrast, the Improved Chaotic Associative Memory for Successive Learning (ICAMSL)[4] and the Divided Chaotic Associative Memory for Successive Learning using Internal Patterns (DCAMSL-IP)[5] have been proposed in order to improve the storage capacity.

In this paper, we propose a Divided Chaotic Associative Memory for Successive Learning (DCAMSL). The proposed model is based on the Improved Chaotic Associative Memory for Successive Learning (ICAMSL)[4] and the Divided Chaotic Associative Memory for Successive Learning using Internal Patterns (DCAMSL-IP)[5]. In the proposed DCAMSL, the learning process and the recall process are not divided. When an unstored pattern set is given to the network, the DCAMSL can learn the patterns successively. Moreover, the storage capacity of the proposed DCAMSL is larger than that of the conventional ICAMSL.

2 Divided Chaotic Associative Memory for Successive Learning

Here, we explain the outline of the proposed Divided Chaotic Associative Memory for Successive Learning (DCAMSL). The proposed DCAMSL has three stages; (1) Pattern Search Stage, (2) Distributed Pattern Generation Stage and (3) Learning Stage.

When an unstored pattern set is given, the DCAMSL recalls the patterns. When an unstored pattern set is given to the network, the DCAMSL changes the internal pattern for input pattern set by chaos and presents other pattern candidates (we call this the Pattern Search Stage). When the DCAMSL can not recall the desired patterns, the distributed pattern is generated by the multi-winners competition[6] (Distributed Pattern Generation Stage), and it learns the input pattern set as an unstored pattern set (Learning Stage).

2.1 Structure of DCAMSL

The proposed DCAMSL is a kind of the hetero-associative memories. Figure 1 shows the structure of the DCAMSL. This model has two layers; an Input/ Output Layer (I/O Layer) composed of conventional neurons and some Distributed Representation Layers (DR Layers) composed of chaotic neurons[7]. In this model, there are the connection weights between neurons in each Distributed Representation Layer and the connection weights between the Input/Output Layer and each Distributed Representation Layer. As shown in Fig.1, the Input/Output Layer has plural parts. The number of parts is decided by depending on the number of patterns included in the pattern set. In the case of Fig.1, the Input/Output Layer consists of N parts corresponding to the patterns 1$\sim N$. In this model, when a pattern set is given to the Input/Output Layer, the internal pattern corresponding to the input patterns is formed in the Distributed Representation Layer. Then, in the Input/Output Layer, an output pattern set is

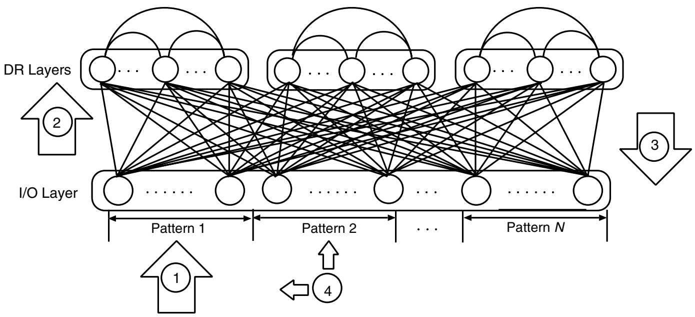


Fig. 1. Structure of Proposed DCAMSL

recalled from the internal pattern. The DCAMSL distinguishes an unstored pattern set from stored patterns by comparing the input patterns with the output patterns.

In this model, the output of the rth Distributed Representation Layer at the time t+1, $x_i^{D_r}(t+1)$ is given by the following equations.

$$x_i^{D_r}(t+1) = \phi^D(\xi_i^r(t+1) + \eta_i^r(t+1) + \zeta_i^r(t+1)) \tag{1}$$

$$\xi_i^r(t+1) = k_s \xi_i^r(t) + \sum_{j=1}^{M} v_{ij}^r A_j(t) \tag{2}$$

$$\eta_i^r(t+1) = k_m \eta_i^r(t) + \sum_{j=1}^{N_r} w_{ij}^r x_j^{D_r}(t) \tag{3}$$

$$\zeta_i^r(t+1) = k_r \zeta_i^r(t) - \alpha_r(t) x_i^{D_r}(t) - \theta_i^r(1-k_r) \tag{4}$$

In Eqs.(1)∼(4), M is the number of neurons in the Input/Output Layer, v_{ij}^r is the connection weight between the neuron j in the Input/Output Layer and the neuron i in the rth Distributed Representation Layer, N_r is the number of neurons in the rth Distributed Representation Layer, w_{ij}^r is the connection weight between the neuron i and the neuron j in the rth Distributed Representation Layer, $\alpha_r(t)$ is the scaling factor of the refractoriness in the rth Distributed Representation Layer at the time t, k_s, k_m and k_r are the damping factors, and θ_i^r is the threshold in the neuron i in the rth Distributed Representation Layer. $\phi^D(\cdot)$ is the following output function:

$$\phi^D(u_i) = \tanh(u_i/\varepsilon) \tag{5}$$

where ε is the steepness parameter.

The output of the neuron j in the Input/Output Layer at the time t, $x_j^{IO_r}(t)$ is given as follows.

$$x_j^{IO_r}(t) = \phi^{IO}\left(\sum_{i=1}^{N_r} v_{ij}^r x_i^{D_r}(t)\right) \tag{6}$$

$$\phi^{IO}(u) = \begin{cases} 1 \;, & u \geq 0 \\ -1 \;, & u < 0 \end{cases} \tag{7}$$

2.2 Pattern Search Stage

In the Pattern Search Stage, when an input pattern set is given, the DCAMSL distinguishes the pattern set from stored patterns. When an unstored pattern set is given, the DCAMSL changes the internal pattern for the input pattern by chaos and presents the other pattern candidates. Until the DCAMSL recalls the desired patterns, the following procedures are repeated. If the DCAMSL can not recall the desired patterns, when the stage is repeated certain times, the DCAMSL finishes the stage.

Pattern Assumption. In the proposed DCAMSL, only when the input patterns are given to all parts of the Input/Output Layer, the patterns are judged. When the input pattern $A_j(t)$ is similar to the recalled pattern $x_j^{IO_r}(t)$, the DCAMSL can assume that input patterns is one of the stored patterns. The DCAMSL outputs the pattern formed by the internal pattern in the Distributed Representation Layer. The similarity rate $s(t)$ is defined by

$$s(t) = \max_r (s_1(t), \cdots, s_r(t), \cdots, s_R(t)) \tag{8}$$

$$s_r(t) = \frac{1}{M}\sum_{j=1}^{M} A_j(t) x_j^{IO_r}(t) \quad (r = 1, \cdots, R) \tag{9}$$

where R is the number of the Distributed Representation Layers. The DCAMSL regards the input patterns as a stored pattern set, when the similarity rate $s_r(t)$ is larger than the threshold $s^{th}(s(t) \geq s^{th})$.

Pattern Search. When the DCAMSL assumes that the input patterns an unstored pattern set, the DCAMSL changes the internal pattern $x_i^{D_r}(t)$ for the input pattern by chaos and presents the other pattern candidates.

In the chaotic neural network, it is known that dynamic association can be realized if the scaling factor of the refractoriness $\alpha_r(t)$ is suitable. Therefore in the proposed model, $\alpha_r(t)$ is changed as follows:

$$\alpha_r(t) = ((\alpha^r_{max}(t) - \alpha_{min})(1 - s_r(t)) + \alpha_{min})/\alpha_{DIV} \tag{10}$$

$$\alpha^r_{max}(t) = M v^r_{max} + N_r w^r_{max} \tag{11}$$

$$v^r_{max} = \max\left(|v^r_{11}|, \cdots, |v^r_{ij}|, \cdots, |v^r_{NM}|\right) \tag{12}$$

$$w^r_{max} = \max\left(|w^r_{11}|, \cdots, |w^r_{ii'}|, \cdots, |w^r_{NN}|\right) \tag{13}$$

where α_{min} is the minimum of α, $\alpha^r_{max}(t)$ is the maximum of α in the rth Distributed Representation Layer at the time t, $s_r(t)$ is the similarity between the input pattern and the output pattern at the time t in the rth Distributed Representation Layer (the time when the Pattern Search Stage started), and α_{DIV} is the constant.

2.3 Distributed Pattern Generation Stage

In the Distributed Pattern Generation Stage, the distributed pattern corresponding to the input patterns is generated by the multi-winners competition[6].

In the proposed DCAMSL, only one distributed pattern is generated in the r^*th Distributed Representation Layer. r^* is decided by

$$r^* = \underset{r}{\operatorname{argmin}} \left(\, |w^1_{max}|, \cdots, |w^r_{max}|, \cdots, |w^R_{max}| \right) \tag{14}$$

where w^r_{max} is the maximum connection weight in rth Distributed Representation Layer, R is the number of the Distributed Representation Layers.

Calculation of Outputs of Neurons in I/O Layer. When the input pattern $A_j(t)$ is given to the Input/Output Layer, the output of the neuron j in the Input/Output Layer x^{IO}_j is given by

$$x^{IO}_j = S_f(A_j(t)) \tag{15}$$

where $S_f(\cdot)$ is the ramp function and is given by

$$S_f(u) = \begin{cases} u, & u > 0 \\ 0, & u \leq 0 \; . \end{cases} \tag{16}$$

Calculation of Initial Output of Neurons in DR Layer. The output of the neuron i in the r^*th Distributed Representation Layer $x^{D_{r^*}(0)}_i$ is calculated by

$$x^{D_{r^*}(0)}_i = \phi^D \left(\sum_{j=1}^{M} v^{r^*}_{ij} \, x^{IO}_j \right) \tag{17}$$

where $v^{r^*}_{ij}$ is the connection weight from the neuron j in the Input/Output Layer to the neuron i in the r^*th Distributed Representation Layer and M is the number of neurons in the Input/Output Layer. The output function $\phi^D(\cdot)$ is given by Eq.(5).

Competition between Neuron in DR Layer. The competition dynamics is given by the following equation:

$$x^{D_{r^*}}_i = \phi^D \left(\sum_{i'=1}^{N_{r^*}} w^{r^*}_{ii'} x^{D_{r^*}}_{i'} \right) \tag{18}$$

where $x^{D_{r^*}}_i$ is the output of the neuron i in the r^*th Distributed Representation Layer and N_{r^*} is the number of neurons in the r^*th Distributed Representation Layer.

2.4 Learning Stage

In the Pattern Search Stage, if the DCAMSL can not recall the desired pattern set, it learns the input pattern set as an unstored pattern set. The Learning Stage has two phases; (1) Hebbian Learning Phase and (2) anti-Hebbian Learning Phase. If the signs of the outputs of two neurons are the same, the connection weight between these two neurons is strengthened. By this learning, the connection weights are changed to learn the input patterns, however the Hebbian learning can only learn a new input pattern set. In the proposed DCAMSL, the anti-Hebbian Learning Phase is employed as similar as the conventional ICAMSL[4]. In the anti-Hebbian Learning Phase, the connection weights are changed in the opposite direction in the case of the Hebbian Learning Phase. The proposed DCAMSL can learn a new pattern set without destroying the stored patterns by the anti-Hebbian learning.

Hebbian Learning Phase. In the Hebbian Learning Phase, until the similarity rate $s(t)$ becomes 1.0, the update of the connection weights is repeated.

The connection weight between the Input/Output Layer and the Distributed Representation Layer $v_{ij}^{r^*}$ and the connection weight in the Distributed Representation Layer $w_{ii'}^{r^*}$ are updated as follows:

$$v_{ij}^{r^*(new)} = v_{ij}^{r^*(old)} + \gamma_v^+ x_i^{D_{r^*}(comp)} A_j(t) \tag{19}$$

$$w_{ii'}^{r^*(new)} = w_{ii'}^{r^*(old)} + \gamma_w^+ x_i^{D_{r^*}(comp)} x_{i'}^{D_{r^*}(comp)} \tag{20}$$

where γ_v^+ is the learning rate of the connection weight $v_{ij}^{r^*}$ in the Hebbian Learning Phase, and γ_w^+ is the learning rate of the connection weight $w_{ii'}^{r^*}$ in this phase.

Give Up Function. When the similarity rate $s(t)$ does not become 1.0 even if the connection weights are updated T_n times, the DCAMSL gives up to memorize the pattern set. If the DCAMSL gives up to learn the pattern set, the anti-Hebbian Learning Phase is not performed.

Anti-Hebbian Learning Phase. The anti-Hebbian Learning Phase is performed after the Hebbian Learning Phase. In this phase, the connection weight $v_{ij}^{r^*}$ and $w_{ii'}^{r^*}$ are changed in the opposite direction in the case of the Hebbian Learning Phase. The anti-Hebbian learning makes the relation between the patterns is learned without destroying the stored patterns.

In this phase, $v_{ij}^{r^*}$ and $w_{ii'}^{r^*}$ are updated by

$$v_{ij}^{r^*(new)} = v_{ij}^{r^*(old)} - \gamma_v^- x_i^{D_{r^*}(comp)} A_j(t) \tag{21}$$

$$w_{ii'}^{r^*(new)} = w_{ii'}^{r^*(old)} - \gamma_w^- x_i^{D_{r^*}(comp)} x_{i'}^{D_{r^*}(comp)} \tag{22}$$

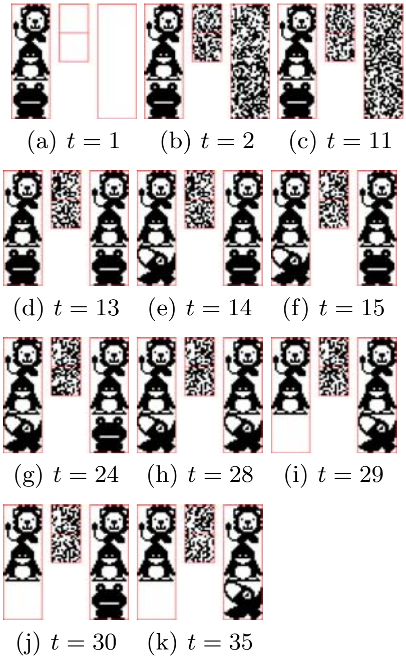
where γ_v^- ($\gamma_v^- > \gamma_v^+ > 0$) is the learning rate of the connection weight $v_{ij}^{r^*}$ in this phase, and γ_w^- ($\gamma_w^- > \gamma_w^+ > 0$) is the learning rate of the connection weight $w_{ii'}^{r^*}$ in this phase.

3 Computer Experiment Results

In this section, we show the computer experiment results to demonstrate the effectiveness of the proposed DCAMSL. The computer experiments were carried out under the conditions shown in Table 1.

Table 1. Experimental Conditions

Learning Parameters		
the number of pattern searches in Pattern Search Stage		10
initial value of all connection weights		$-1.0 \sim 1.0$
learning rate in Hebbian Learning	γ_v^+ , γ_w^+	1.0
learning rate in anti-Hebbian Learning	γ_v^- , γ_w^-	2.0
threshold of similarity rate	s^{th}	1.0
Chaotic Neuron Parameters		
constant for refractoriness	α_{DIV}	25
minimum of scaling factor α	α_{min}	0.0
damping factor	k_s	0.5
damping factor	k_m	0.3
damping factor	k_r	0.95
threshold of neurons	θ_i^r	0.0
steepness parameter	ε	0.0005
Competition Parameters		
steepness parameter	ε	0.0005
the number of competitions	T	50


(a) $t = 1$ (b) $t = 2$ (c) $t = 11$

(d) $t = 13$ (e) $t = 14$ (f) $t = 15$

(g) $t = 24$ (h) $t = 28$ (i) $t = 29$

(j) $t = 30$ (k) $t = 35$

Fig. 2. Successive Learning in Proposed Model

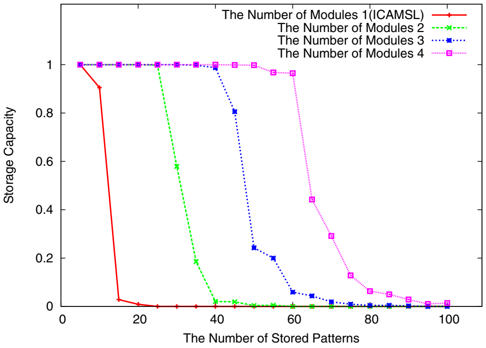


Fig. 3. Storage Capacity

3.1 Successive Learning and One-to-Many Associations

Figure 2 shows the successive learning and one-to-many associations in the proposed DCAMSL. As seen in Fig.2, the patterns "lion", "penguin" and "frog" were given to the network at t=1. At t=1, the DCAMSL could not recall the correct patterns because no pattern was stored in the network. During t=2∼11, the DCAMSL changed the internal patterns by chaos and presented the other patterns. As a result, the DCAMSL regarded the input patterns as unstored patterns, at t=13, the patterns "lion", "penguin" and "frog" were trained as new patterns.

At t=14, the patterns "lion", "penguin" and "crow" were given to the network. At this time, since only the pattern set "lion", "penguin" and "frog" was memorized in the network, the DCAMSL recalled the patterns "lion", "penguin" and "frog". During t=15∼24, the DCAMSL changed the internal patterns by chaos and presented the other pattern candidates, however it could not recall the correct patterns as unstored patterns, at t=28, the "lion", "penguin" and "crow" were trained as new patterns.

At t=29, the patterns "lion" and "penguin" were given to the network, the DCAMSL recalled "lion", "penguin" and "frog" (t=30) and "lion", "penguin" and "crow" (t=35). From these results, we confirmed that the proposed DCAMSL can learn patterns successively and realize one-to-many associations.

3.2 Storage Capacity

Here, we examined the storage capacity of the proposed DCAMSL. In this experiment, we used the DCAMSL which has 200 neurons (100 neurons for pattern 1 and 100 neurons for pattern 2) in the Input/Output Layer and 420 neurons in the Distributed Representation Layer. We used random patterns to store and Fig.3 shows the average of 100 trials. In this figure, the horizontal axis is the number of stored pattern pairs, and the vertical axis is the perfect recall rate. In this figure, the storage capacity of the Improved Chaotic Associative Memory for Successive Learning (ICAMSL)[4] is also shown for reference. From these results, we confirmed that the storage capacity of the proposed DCAMSL is larger than that of the conventional ICAMSL.

4 Conclusions

In this paper, we have proposed the Divided Chaotic Associative Memory for Successive Learning (DCAMSL). The proposed model is based on the Improved Chaotic Associative Memory for Successive Learning (ICAMSL)[4] and the Divided Chaotic Associative Memory for Successive Learning using Internal Patterns (DCAMSL-IP). In the proposed DCAMSL, the learning process and recall process are not divided. When an unstored pattern set is given to the network, the DCAMSL can learn the pattern successively. We carried out a series of computer experiments and confirmed that the proposed DCAMSL can learn patterns successively and realize one-to-many associations, and the storage capacity of the DCAMSL is larger than that of the conventional ICAMSL.

References

1. Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences U.S.A. 79, 2554–2558 (1982)
2. Kosko, B.: Bidirectional associative memory. IEEE Transactions on Neural Networks 18(1), 49–60 (1988)
3. Carpenter, G.A., Grossberg, S.: Pattern Recognition by Self-organizing Neural Networks. MIT Press, Cambridge (1995)
4. Ikeya, T., Sazuka, T., Hagiwara, A., Osana, Y.: Improved chaotic associative memory for successive learning. In: Proceedings of IASTED Artificial Intelligence and Applications, Innsbruck (2007)
5. Kawasaki, N., Osana, Y., Hagiwara, M.: Divided chaotic associative memory for successive learning using internal patterns. In: Proceedings of 7th International Conference on Neural Information Processing, Taejon (2000)
6. Huang, J.T., Hagiwara, M.: A multi-winners self-organizing neural network. In: IEEE International Conference on System, Man and Cybernetics, pp. 2499–2504 (1997)
7. Aihara, K., Takabe, T., Toyoda, M.: Chaotic neural networks. Physics Letter A 144(6,7), 333–340 (1990)

Reinforcement Learning Using Kohonen Feature Map Associative Memory with Refractoriness Based on Area Representation

Atsushi Shimizu and Yuko Osana

Tokyo University of Technology,
1401-1 Katakura, Hachioji, Tokyo, Japan
osana@cc.teu.ac.jp

Abstract. In this paper, we propose a reinforcement learning method using Kohonen Feature Map Associative Memory with Refractoriness based on Area Representation. The proposed method is based on the actor-critic method, and the actor is realized by the Kohonen Feature Map Associative Memory with Refractoriness based on Area Representation. The Kohonen Feature Map Associative Memory with Refractoriness based on Area Representation is based on the self-organizing feature map, and it can realize successive learning and one-to-many associations. Moreover, it has robustness for noisy input and damaged neurons because it is based on the area representation. The proposed method makes use of this property in order to realize the learning during the practice of task. We carried out a series of computer experiments, and confirmed the effectiveness of the proposed method in path-finding problem.

1 Introduction

The reinforcement learning is a sub-area of machine learning concerned with how an agent ought to take actions in an environment so as to maximize some notion of long-term reward[1]. Reinforcement learning algorithms attempt to find a policy that maps states of the world to the actions the agent ought to take in those states.

Temporal Difference (TD) learning is one of the reinforcement learning algorithm. The TD learning is a combination of Monte Carlo ideas and dynamic programming (DP) ideas. TD resembles a Monte Carlo method because it learns by sampling the environment according to some policy. TD is related to dynamic programming techniques because it approximates its current estimate based on previously learned estimates. The actor-critic method[2] is the method based on the TD learning, and consists of two parts; (1) actor which selects the action and (2) critic which evaluate the action and the state.

On the other hand, neural networks are drawing much attention as a method to realize flexible information processing. Neural networks consider neuron groups of the brain in the creature, and imitate these neurons technologically. Neural networks have some features, especially one of the important features is that the networks can learn to acquire the ability of information processing. The flexible

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 212–219, 2009.

information processing ability of the neural network and the adaptive learning ability of the reinforcement learning are combined, some reinforcement learning method using neural networks are proposed[3][4].

In this paper, we propose the reinforcement learning method using Kohonen Feature Map Associative Memory with Refractoriness based on Area Representation(KFMAM-R-AR)[5]. The proposed method is based on the actor-critic method, and the actor is realized by the KFMAM-R-AR. The KFMAM-R-AR is based on the self-organizing feature map[6], and it can realize successive learning and one-to-many associations. The proposed method makes use of this property in order to realize the learning during the practice of task.

2 Kohonen Feature Map Associative Memory with Refractoriness Based on Area Representation

Here, we explain the conventional Kohonen Feature Map Associative Memory with Refractoriness based on Area Representation (KFMAM-R-AR)[5] which can realize one-to-many associations. This model is based on the Kohonen Feature Map with Area Representation (KFMAM-AR)[7], and refractoriness is introduced to the neurons in the Map-Layer. In the KFMAM-R-AR, one-to-many associations are realized by the refractoriness of neurons. And, in the model, enough robustness for damaged neurons when analog patterns are memorized are realized by improvement of the calculation of the internal states of neurons in the Map-Layer.

2.1 Structure

Figure 1 shows the structure of the KFMAM-R-AR. As shown in Fig.1, the KFMAM-R-AR has two layers; (1) Input/Output(I/O)-Layer and (2) Map-Layer, and the I/O-Layer is divided into some parts.

In the KFMAM-R-AR, since one concept is expressed by the winner neuron and some neurons located adjacent to the winner neuron, it has the robustness for damaged neurons in the Map-Layer.

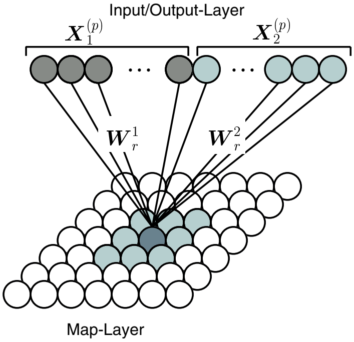


Fig. 1. Structure of KFMAM-R-AR

2.2 Learning Process

The learning algorithm for the KFMAM-R-AR is based on the conventional sequential learning algorithm for the KFMAM-AR.

In the sequential learning algorithm for the KFMAM-R-AR, the connection weights are learned as follows:

(1) The initial values of weights are chosen randomly.

(2) The Euclid distance between the learning vector $\boldsymbol{X}^{(p)}$ and the connection weights vector $\boldsymbol{W}_i$, $d(\boldsymbol{X}^{(p)}, \boldsymbol{W}_i)$ is calculated.

$$d(\boldsymbol{X}^{(p)}, \boldsymbol{W}_i) = \sqrt{\sum_{k=1}^{M} (X_k^{(p)} - W_{ik})^2} \tag{1}$$

(3) The winner neuron r whose Euclid distance is minimum is found.

$$r = \underset{i}{\text{argmin}}\, d(\boldsymbol{X}^{(p)}, \boldsymbol{W}_i) \tag{2}$$

(4) If $d(\boldsymbol{X}^{(p)}, \boldsymbol{W}_i) > \theta^l$, the connection weights of the winner neuron r are fixed. The connection weights except those of fixed neurons are changed by

$$\boldsymbol{W}_i(t+1) = \boldsymbol{W}_i(t) + H(d_i)\alpha(t)h_{ri}(\boldsymbol{X}^{(p)} - \boldsymbol{W}_i(t)) \tag{3}$$

where h_{ri} is the neighborhood function and is given by

$$h_{ri} = \exp\left(\frac{-||\boldsymbol{r} - \boldsymbol{i}||^2}{2\sigma(t)^2}\right) \tag{4}$$

where $\sigma(t)$ is the following decreasing function:

$$\sigma(t) = \sigma_i \left(\frac{\sigma_f}{\sigma_i}\right)^{t/T}. \tag{5}$$

In this equation, σ_i is the initial value of $\sigma(t)$ and $\sigma(t)$ varies from σ_i to $\sigma_f (\sigma_i > \sigma_f)$. T is the upper limit of the learning iterations.
In Eq.(3), $\alpha(t)$ is the learning rate and is given by:

$$\alpha(t) = \frac{-\alpha_0(t-T)}{T} \tag{6}$$

where α_0 is the initial value of $\alpha(t)$, and $H(d_i)$ is calculated by

$$H(d_i) = \frac{1}{1 + \exp(-(d_i - D)/\varepsilon).} \tag{7}$$

In this equation, d_i is the Euclid distance between the neuron i and the nearest weights fixed neuron in the Map-Layer, D is the constant and ε is the steepness parameter of the function $H(d_i)$. Owing to $H(d_i)$, weights of neurons close to the fixed neurons are semi-fixed, that is, they become hard to be learned.

(5) (2)∼(4) are iterated until $d(\boldsymbol{X}^{(p)}, \boldsymbol{W}_i) \leq \theta^l$ is satisfied.

(6) The connection weights of the winner neuron r, $\boldsymbol{W}_r$ are fixed.

(7) (2)∼(6) are iterated when a new pattern set is given.

2.3 Recall Process

In the recall process of the KFMAM-R-AR, when the pattern $\boldsymbol{X}$ is given to the I/O-Layer, the output of the neuron i in the Map-Layer at the time t, $x_i^{map}(t)$ is calculated by

$$x_i^{map}(t) = H^{recall}(d(\boldsymbol{r},\boldsymbol{i}))f(u_i^{map}(t)) \tag{8}$$

$$H^{recall}(d(\boldsymbol{r},\boldsymbol{i})) = \frac{1}{1+\exp\left((d(\boldsymbol{r},\boldsymbol{i})-D)/\varepsilon\right)} \tag{9}$$

where D is the constant which decides the area size, ε is the steepness parameter. $d(\boldsymbol{r},\boldsymbol{i})$ is the Euclid distance between the winner neuron r and the neuron i and is calculated by

$$r = \underset{i}{\operatorname{argmax}}\, u_i^{map}(t). \tag{10}$$

Owing to $H^{recall}(d(\boldsymbol{r},\boldsymbol{i}))$, which are far from the winner neuron become hard fire. $f(u_i^{map}(t))$ is calculated by

$$f(u_i^{map}(t)) = \begin{cases} 1, & \text{if } u_i^{map}(t) > \theta^{map} \text{ and } u_i^{map}(t) > \theta^{min} \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

where $u_i^{map}(t)$ is the internal state of the neuron i in the Map-Layer at the time t, θ^{map} and θ^{min} are the threshold of the neuron in the Map-Layer. θ^{map} is calculated as follows:

$$\theta^{map} = \min_i(u_i^{map}(t)) + a(\max_i(u_i^{map}(t)) - \min_i(u_i^{map}(t))) \tag{12}$$

where a $(0.5 < a < 1)$ is the coefficient.

In Eq.(8), when the binary pattern $\boldsymbol{X}$ is given to the I/O-Layer, the internal state of the neuron i in the Map-Layer $u_i^{map}(t)$ is calculated by

$$u_i^{map}(t) = 1 - \frac{d^{in}(\boldsymbol{X},\boldsymbol{W}_i)}{\sqrt{N^{in}}} - \alpha\sum_{d=0}^{t} k_r^d x_i^{map}(t-d) \tag{13}$$

where $d^{in}(\boldsymbol{X},\boldsymbol{W}_i)$ is the Euclid distance between the input patterns $\boldsymbol{X}$ and the connection weights $\boldsymbol{W}_i$. In the recall process, since all neurons in the I/O-Layer not always receive the input, the distance for the part where the pattern is given is calculated by

$$d^{in}(\boldsymbol{X},\boldsymbol{W}_i) = \sqrt{\sum_{k\in C}(X_k - W_{ik})^2} \tag{14}$$

where C shows the set of the neurons in the I/O-Layer which receive the input. In Eq.(13), N^{in} is the number of neurons which receive the input in the I/O-Layer, α is the scaling factor of the refractoriness, $k_r (0 \leq k_r < 1)$ is the damping factor. The output of the neuron k in the I/O-Layer $x_k^{in}(t)$ is calculated by

$$x_k^{in}(t) = \begin{cases} 1, & \text{if } u_k^{in}(t) \geq \theta_b^{in} \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

$$u_k^{in}(t) = \frac{1}{\sum\limits_{i:x_i>\theta^{out}} x_i^{map}(t)} \sum_{i:x_i>\theta^{out}} W_{ik} \tag{16}$$

where θ_b^{in} is the threshold of the neuron in the I/O-Layer, θ^{out} is the threshold for the output of the neuron in the Map-Layer.

3 Reinforcement Learning Using KFMAM-R-AR

Here, we explain the proposed reinforcement learning method using Kohonen Feature Map Associative Memory with Refractoriness based on Area Representation.

Outline In the proposed method, the actor in the Actor-Critic[2] is realized by the KFMAM-R-AR. In this research, the I/O-Layer in the KFMAM-R-AR is divided into two parts corresponding to the state $\boldsymbol{s}$ and the action $\boldsymbol{a}$, and the actions for the states are memorized(Fig.2).

In this method, the critic receives the states which are obtained from the environment, the state is estimated and the value function is updated. Moreover, the critic outputs Temporal Difference(TD) error to the actor. The KFMAM-R-AR which behaves as the actor (we call this "actor network") is trained based on the TD error, and selects the action from the state of environment. Figure 3 shows the flow of the proposed method.

3.1 Actor Network

In the proposed method, the actor in the Actor-Critic[2] is realized by the KFMAM-R-AR.

Dynamics. In the actor network, when the state $\boldsymbol{s}$ is given to the I/O-Layer, the corresponding action $\boldsymbol{a}$ is recalled.

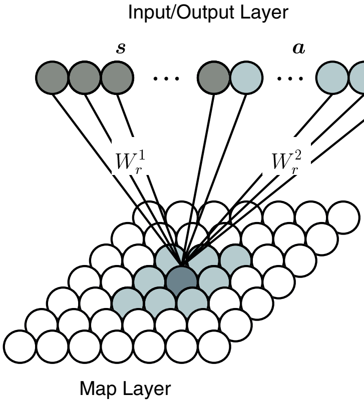


Fig. 2. Structure of Actor Network

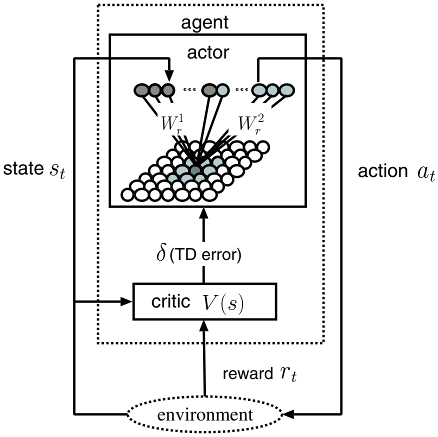


Fig. 3. Flow of Proposed Method

When the pattern $\boldsymbol{X}$ is given to the network, the output of the neuron i in the Map-Layer at the time t $x_i^{map}(t)$ is given by Eq.(11). In the actor network, only the state information is given, so the input pattern is given by

$$\boldsymbol{X} = (\boldsymbol{s}(t), \boldsymbol{0})^T \tag{17}$$

where $\boldsymbol{s}(t)$ is the state at the time t.

The internal state of the neuron i in the Map-Layer at the time t $u_i^{map}(t)$ is given by Eq.(13).

The output of the neuron k in the I/O-Layer at the time t $x_k^{in}(t)$ is given by Eq.(15).

Learning. The actor network is trained based on the TD error from the critic.

The learning vector at the time t $\boldsymbol{X}^{(t)}$ is given by the state $\boldsymbol{s}(t)$ and the corresponding action $\boldsymbol{a}(t)$ as follows.

$$\boldsymbol{X}^{(t)} = (\boldsymbol{s}(t), \boldsymbol{a}(t))^T \tag{18}$$

The learning vector $\boldsymbol{X}^{(t)}$ is trained the following procedures.

(1) The Euclid distance between the learning vector $\boldsymbol{X}^{(t)}$ and the weight vector $\boldsymbol{W}_i(t)$, $d(\boldsymbol{X}^{(t)}, \boldsymbol{W}_i(t))$ is calculated by

$$d(\boldsymbol{X}^{(t)}, \boldsymbol{W}_i(t)) = \sqrt{\sum_{k=1}^{M} (X_k^{(t)} - W_{ik}(t))^2} \tag{19}$$

where M is the number of neurons in the I/O-Layer.

(2) The winner neuron r whose Euclid distance is minimum is found.

$$r = \underset{i}{\operatorname{argmin}}\, d(\boldsymbol{X}^{(t)}, \boldsymbol{W}_i(t)) \tag{20}$$

(3) The connection weights are updated based on the TD error.

(3-1) When TD error is smaller than 0

When the undesired action $\boldsymbol{a}(t)$ for the state $\boldsymbol{s}(t)$ is selected, the connection weights are updated so that the $\boldsymbol{X}^{(t)}$ is not recalled. If the connection weights for the winner neuron r selected in (2) are fixed, they are unlocked.

The connection weighs are updated as follows.

$$\boldsymbol{W}_i(t+1) = \boldsymbol{W}_i(t) + \delta\beta(1 - H(d_i))h_{ri}(\boldsymbol{X}^{(t)} - \boldsymbol{W}_i(t)) \tag{21}$$

where δ is TD error, and β is random number. $H(\cdot)$ the semi-fix function given by Eq.(7). And h_{ri} is the neighborhood function give by Eq.(4).

(3-2) When TD error is greater than 0

When the desired action $\boldsymbol{a}(t)$ for the state $\boldsymbol{s}(t)$ is selected, the learning vector $\boldsymbol{X}^{(t)}$ is memorized.

The connection weights are updated as follows.

$$\boldsymbol{W}_i(t+1) = \boldsymbol{W}_i(t) + \delta\beta H(d_i)h_{ri}(\boldsymbol{X}^{(t)} - \boldsymbol{W}_i(t)) \tag{22}$$

If $d(\boldsymbol{X}^{(t)}, \boldsymbol{W}_r) \leq \theta^l$, the connection weights of the winner neuron r are fixed.

(3-3) When TD error is zero

When the TD error is zero, the connection weights are not updated.

3.2 Reinforcement Learning Using KFMAM-R-AR

The flow of the proposed reinforcement learning method using KFMAM-R-AR is as follows:

(1) The initial values of weights in the actor network are chosen randomly.
(2) The agent observes the environment $\boldsymbol{s}(t)$, and the actor $\boldsymbol{a}(t)$ is selected by the actor network.
(3) The state $\boldsymbol{s}(t)$ transits to the $\boldsymbol{s}(t+1)$ by action $\boldsymbol{a}(t)$.
(4) The critic receives the reward $r(\boldsymbol{s}(t+1))$ from the environment $\boldsymbol{s}(t+1)$, and outputs the TD error δ to the actor.

$$\delta = r(\boldsymbol{s}(t+1)) + \gamma V(\boldsymbol{s}(t+1)) - V(\boldsymbol{s}(t)) \tag{23}$$

where γ $(0 \leq \gamma \leq 1)$ is the decay parameter, $V(\boldsymbol{s}(t))$ is the value function for the state $\boldsymbol{s}(t)$.
(5) The eligibility $e_t(\boldsymbol{s})$ is updated.

$$e(\boldsymbol{s}) \leftarrow \begin{cases} \gamma\lambda e(\boldsymbol{s}) & (\text{if } \boldsymbol{s} \neq \boldsymbol{s}(t+1)) \\ \gamma\lambda e(\boldsymbol{s}) + 1 & (\text{if } \boldsymbol{s} = \boldsymbol{s}(t+1)) \end{cases} \tag{24}$$

where γ $(0 \leq \gamma \leq 1)$ is the decay parameter, and λ is the trace decay parameter.
(6) All values for states $V(\boldsymbol{s})$ are updated based on the eligibility $e_t(\boldsymbol{s})$ $(\boldsymbol{s} \in S)$.

$$V(\boldsymbol{s}) \leftarrow V(\boldsymbol{s}) + \xi\delta e_t(\boldsymbol{s}) \tag{25}$$

where ξ $(0 \leq \xi \leq 1)$ is the learning rate.
(7) The connection weights in the actor network are updated based on the TD error (See 3.1).
(8) Back to (2).

4 Computer Experiment Results

Here, we show the computer experiment results to demonstrate the effectiveness of the proposed method.

We applied the proposed method to the path-finding problem. In this experiment, a agent moves from the start point (S) to the goal point (G). The agent can observe the states of six cells in the lattice including the agent, and can move forward/backward/left/right. As the positive reward, we gave 0.5 when the agent arrived at the goal, and as the negative reward, we gave −0.045 when the agent hit against the wall and for every actions.

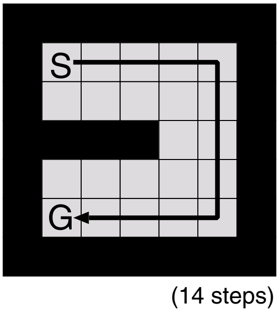


Fig. 4. Trained Route

Figure 4 shows an example of map and the trained route(arrow). Figure 5 shows an example of trained relation between the state and the action. Figure 6 shows the transition of number of average/minimum steps from the start to the goal in the same trial.

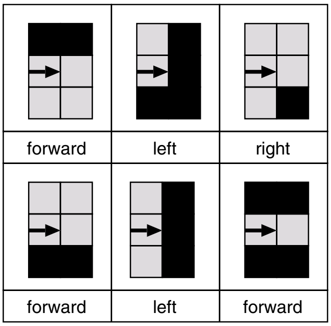


Fig. 5. An example of Trained Relation between State and Action

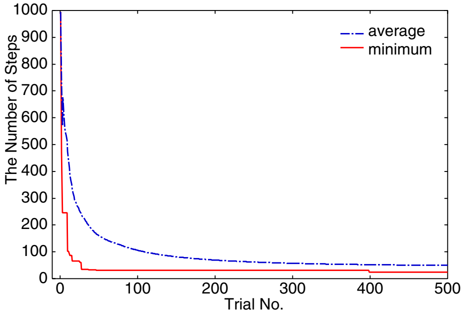


Fig. 6. Transition of Steps

We carried out the similar experiments using other maps, and confirmed that the proposed method can learn the path from the start to the goal.

5 Conclusion

In this paper, we have proposed the reinforcement learning method using Kohonen Feature Map Associative Memory with Refractoriness based on Area Representation. The proposed method is based on the actor-critic method, and the actor is realized by the KFMAM-R-AR. We carried out a series of computer experiments, and confirmed the effectiveness of the proposed method in path-finding problem.

References

1. Sutton, R.S., Barto, A.G.: Reinforcement Learning, An Introduction. MIT Press, Cambridge (1998)
2. Witten, I.H.: An adaptive optimal controller for discrete-time Markov environments. Information and Control 34, 286–295 (1977)
3. Shibata, K., Sugisaka, M., Ito, K.: Fast and stable learning in direct-vision-based reinforcement learning. In: Proceedings of the 6th International Sysmposium on Artificial Life and Robotics, vol. 1, pp. 200–203 (2001)
4. Ishii, S., Shidara, M., Shibata, K.: A model of emergence of reward expectancy neurons by reinforcement learning. In: Proceedings of the 10th International Sysmposium on Artificial Life and Robotics, vol. GS21-5 (2005)
5. Imabayashi, T., Osana, Y.: Implementation of association of one-to-many associations and the analog pattern in Kohonen feature map associative emory with area representation. In: Proceedings of IASTED Artificial Intelligence and Applications, Innsbruck (2008)
6. Kohonen, T.: Self-Organizing Maps. Springer, Heidelberg (1994)
7. Abe, H., Osana, Y.: Kohonen feature map associative memory with area representation. In: Proceedings of IASTED Artificial Intelligence and Applications, Innsbruck (2006)

Automatic Model Selection via Corrected Error Backpropagation

Masashi Sekino and Katsumi Nitta

Tokyo Institute of Technology,
4259-J2-53 Nagatsuta-cho, Midori-ku, Yokohama, 226-8502 Japan
{sekino,nitta}@ntt.dis.titech.ac.jp
http://www.ntt.dis.titech.ac.jp/~sekino

Abstract. In this paper, we propose a learning method called *Corrected Error Backpropagation* which maximizes the corrected log-likelihood which works like Akaike Information Criterion. For the purpose of maximizing the corrected log-likelihood, we introduce temperature parameter for the corrected log-likelihood. This paper also shows an optimal scheduling of the temperature parameter. Applying to our method to a linear regression model on the Boston house price estimation problem and multi layered perceptrons on the DELVE datasets, the method gives good results.

1 Introduction

When a learning model is redundant or there is not enough learning samples, maximum likelihood estimation gives an overfitted estimate. Therefore, in practice, we should select an appropriate learning model from some learning models after they are estimated by maximum likelihood estimation. Overfitting process, which is gradual increase of the generalization error in the learning process of a sequential learning method such as error backpropagation (BP) [1], is also observed in the learning of a redundant neural network.

In unidentifiable cases, where the true function has smaller rank than the learning model, Fukumizu [2] showed that overfitting occurs when a linear neural network is trained by BP. Fukumizu [3] also showed that the generalization error of the maximum likelihood estimator (MLE) in the unidentifiable cases is bigger than that in the identifiable cases, and that the generalization error of the MLE in the almost unidentifiable cases where the true function has very small singular values is close to that in the unidentifiable cases.

There is early-stopping [4] for dealing with the overfitting process and regularization for dealing with the overfitting. Here, we need to select a stopping time or a regularization parameter appropriately. Therefore, even if we apply these methods, the necessity of the selection is unchanged.

In this paper, we propose a learning method called *Corrected Error Backpropagation* which maximizes the corrected log-likelihood which works like Akaike Information Criterion [5]. For the purpose of maximizing the corrected

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 220–227, 2009.

log-likelihood, we introduce temperature parameter for the corrected log-likelihood. This paper also shows an optimal scheduling of the temperature parameter. The results of applications to our method to a linear regression model on the Boston house price estimation problem and multi layered perceptrons on the DELVE datasets are also shown.

2 Preliminaries

2.1 Regression Problems

We consider a set of functions $\{f(\boldsymbol{x};\boldsymbol{\vartheta})\}$ from $\boldsymbol{x} \in \mathbb{R}^M$ to $y \in \mathbb{R}$ as a regression model, and a function $f(\boldsymbol{x};\boldsymbol{\vartheta}_0)$ in the regression model as a true function. We assume $(\boldsymbol{x}, y)$ is independently generated from $q(\boldsymbol{x})$ and $p(y|\boldsymbol{x};\boldsymbol{\theta}_0) = \mathcal{N}(y; f(\boldsymbol{x};\boldsymbol{\vartheta}_0), \sigma_0^2)$, and we adopt $p(y|\boldsymbol{x};\boldsymbol{\theta})$ as a learning model. $\mathcal{N}(\cdot;\mu,\sigma^2)$ is the normal distribution with a mean μ and a variance σ^2, $\boldsymbol{\theta} = (\boldsymbol{\vartheta}, \sigma^2)$ and $\boldsymbol{\theta}_0 = (\boldsymbol{\vartheta}_0, \sigma_0^2)$.

Log-empirical likelihood to learning samples $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ is

$$\mathcal{L}(\boldsymbol{\theta};\mathcal{D}) = \log q(\mathcal{D}) - \frac{N}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}J(\boldsymbol{\vartheta};\mathcal{D}), \tag{1}$$

$$J(\boldsymbol{\vartheta};\mathcal{D}) = \sum_{n=1}^{N}\big(y_n - f(\boldsymbol{x}_n;\boldsymbol{\vartheta})\big)^2. \tag{2}$$

$J(\boldsymbol{\vartheta};\mathcal{D})$ is called squared empirical error.

2.2 Neural Networks

Three layered neural network is defined as

$$f(\boldsymbol{x};\boldsymbol{\vartheta}) = \boldsymbol{\phi}(\boldsymbol{x};\boldsymbol{\vartheta})^T\boldsymbol{w}, \tag{3}$$

$$\boldsymbol{\phi}(\boldsymbol{x};\boldsymbol{\vartheta}) = (\phi(\boldsymbol{x};\boldsymbol{\varphi}_1), \cdots, \phi(\boldsymbol{x};\boldsymbol{\varphi}_H))^T, \tag{4}$$

where ^{T} denotes a transpose, $\boldsymbol{w} \in \mathbb{R}^H$ is a coefficient vector, $\phi(\boldsymbol{x};\boldsymbol{\varphi}_i)$ is a basis function which is defined by a parameter vector $\boldsymbol{\varphi}_i \in \mathbb{R}^L$, and $\boldsymbol{\vartheta} = (w_1, \cdots, w_H, \boldsymbol{\varphi}_1^T, \cdots, \boldsymbol{\varphi}_H^T)^T$ is a parameter vector of the neural network. The number of basis functions H is called size of the neural network. The parameter vector of the learning model $p(y|\boldsymbol{x};\boldsymbol{\theta})$ is $\boldsymbol{\theta} = (\boldsymbol{\vartheta}^T, \sigma^2)^T$.

Hyperbolic tangent:

$$\phi(\boldsymbol{x};\boldsymbol{\varphi}_i) = \tanh(\boldsymbol{x}^T\boldsymbol{\varphi}_i) \tag{5}$$

is usually used for multi layered perceptrons, We usually set x_1 as $x_1 = 1$ and use φ_1 as an intercept.

3 Corrected Error Backpropagation

3.1 Corrected Log-Likelihood

Akaike Information Criterion (AIC) [5] is defined as

$$\mathrm{IC}(\mathcal{M};\mathcal{D}) = -2\mathcal{L}(\hat{\boldsymbol{\theta}};\mathcal{D}) + 2A\cdot\dim(\boldsymbol{\theta}). \tag{6}$$

$\mathrm{IC}(\mathcal{M};\mathcal{D})$ becomes AIC when $A = 1$. $\hat{\boldsymbol{\theta}}$ is the maximum likelihood estimate.

Because linear regression model or neural network contains the model which has only subset of the variables or the basis functions of it, these models are called hierarchical models. Therefore, there exists a sub parameter space corresponding to the size H' ($H' \leq H$) model on the parameter space of the size H model. When the minimum size of the model which realizes the same distribution of a parameter $\boldsymbol{\theta}$ is H', we use the effective size $\mathrm{size}(\boldsymbol{\theta})_{eff}$ as $\mathrm{size}(\boldsymbol{\theta})_{eff} = H'$. Then, we can count the effective number of parameters for the parameter $\boldsymbol{\theta}$ as

$$\dim(\boldsymbol{\theta})_{eff} = \mathrm{size}(\boldsymbol{\theta})_{eff}(L+1) + 1. \tag{7}$$

Here, $L+1$ is for the parameter vector $\boldsymbol{\varphi}_i$ and coefficient value w_i. The second term 1 is for the variance parameter σ^2.

Using the effective number of parameters, we define corrected log-likelihood as

$$\mathcal{L}_C(\boldsymbol{\theta};\mathcal{D}) = \mathcal{L}(\boldsymbol{\theta};\mathcal{D}) - A\cdot\dim(\boldsymbol{\theta})_{eff}. \tag{8}$$

While the information criterion (6) evaluates the maximum likelihood estimate $\hat{\boldsymbol{\theta}}$, the corrected log-likelihood evaluates parameter $\boldsymbol{\theta}$.

Because maximization of the corrected log-likelihood under fixed $\dim(\boldsymbol{\theta})_{eff}$ is mere maximum likelihood estimation under $\dim(\boldsymbol{\theta})_{eff}$, maximization of the corrected log-likelihood without fixing $\dim(\boldsymbol{\theta})_{eff}$ is selection of the optimal size $\hat{H}^*$ which minimizes the information criterion (6) and maximum likelihood estimation of the size $\hat{H}^*$ model.

3.2 Corrected Error Backpropagation

We call the gradient maximization of the corrected log-likelihood *Corrected Error Backpropagation*. Corrected error backpropagation modify the parameter as:

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) + \eta \left.\frac{\partial \mathcal{L}_C(\boldsymbol{\theta};\mathcal{D})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}(t)}, \tag{9}$$

where t is a discrete time and η is a learning rate.

3.3 Annealing Optimization

The corrected term of the corrected log-likelihood (8) cannot work appropriately because $\mathrm{size}(\boldsymbol{\theta})_{eff}$ has discrete shape on the parameter space. Therefore, we

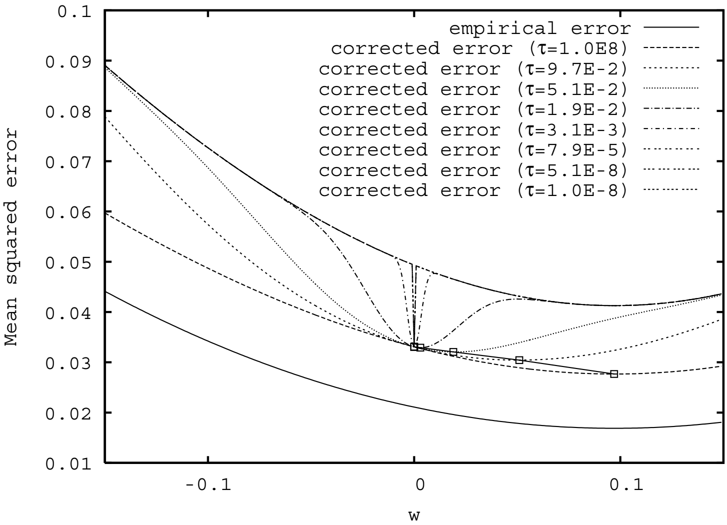


Fig. 1. Approximated corrected error ($A = 1$, $N = 5$)

approximate size$(\boldsymbol{\theta})_{eff}$ by a smooth function and gradually make it closer to size$(\boldsymbol{\theta})_{eff}$. We define the approximated effective size as

$$\mathrm{size}(\boldsymbol{\theta};\tau)_{eff} = \sum_{i=1}^{H} g(w_i;\tau), \tag{10}$$

$$g(x;\tau) = 1 - \exp\left(-\frac{x^2}{2\tau^2}\right) \tag{11}$$

where τ is a temperature parameter. If there is no i which satisfies $\phi(\boldsymbol{x};\boldsymbol{\varphi}_i) = 0$ and there are no i and j ($i \neq j$) which satisfy $\phi(\boldsymbol{x};\boldsymbol{\varphi}_i) = \phi(\boldsymbol{x};\boldsymbol{\varphi}_j)$ or $\phi(\boldsymbol{x};\boldsymbol{\varphi}_i) = -\phi(\boldsymbol{x};\boldsymbol{\varphi}_j)$, size$(\boldsymbol{\theta};\tau)_{eff}$ goes size$(\boldsymbol{\theta})_{eff}$ when $\tau \to 0$.

We use the approximated number of parameters $\dim(\boldsymbol{\theta};\tau)_{eff}$ using size$(\boldsymbol{\theta};\tau)_{eff}$ for size$(\boldsymbol{\theta})_{eff}$ in (7). And we use the approximated corrected log-likelihood $\mathcal{L}_C(\boldsymbol{\theta};\mathcal{D},\tau)$ using $\dim(\boldsymbol{\theta};\tau)_{eff}$ for $\dim(\boldsymbol{\theta})_{eff}$ in (8).

Fig. 1[1] shows the approximated corrected squared error:

$$J_C(w;\mathcal{D},\tau) = \exp\left(-\frac{2}{N}\bar{\mathcal{L}}_C((w,\hat{\sigma}^2)^T;\mathcal{D},\tau)\right) \tag{12}$$

which is a converted function of the approximated corrected log-likelihood of a linear regression model $f(x;w) = wx$ when a true function is $f_0(x) = 0$:

$$\mathcal{L}_C((w,\hat{\sigma}^2)^T;\mathcal{D},\tau) = \bar{\mathcal{L}}_C((w,\hat{\sigma}^2)^T;\mathcal{D},\tau) + Const., \tag{13}$$

[1] $a\mathrm{E}b$ stands for $a \times 10^b$.

$$\bar{\mathcal{L}}_C((w, \hat{\sigma}^2)^T; \mathcal{D}, \tau) = -\frac{N}{2} \log J(w; \mathcal{D}) - A \cdot \dim(\boldsymbol{\theta}; \tau)_{eff}. \qquad (14)$$

In this case, AIC would eliminate the parameter w because the approximated corrected squared error with $\tau = 1.0 \times 10^{-8}$ is minimized at $w = 0$. If the initial value of w is negative, we can obtain the optimal value $w = 0$ by minimization of the approximated corrected squared error with $\tau = 1.0E \times 10^{-8}$ using gradient descent. However, if the initial value of w is positive, gradient descent gives $w \approx 0.1$ which is the minimizer of the empirical squared error. The approximated corrected squared error with $\tau = 1.0 \times 10^{8}$ has almost the same shape as the empirical squared error. Therefore the minimizer of the approximated corrected squared error with $\tau = 1.0 \times 10^{8}$ is $w \approx 0.1$. If this value is set to the initial value of w, gradient descent gives smaller w by minimization of the approximated corrected squared error with $\tau = 9.7 \times 10^{-2}$. By repeating minimization of the approximated corrected squared error with fixed τ and reduction of τ, we can obtain $w = 0$ as shown in Fig. 1

There is a similar approach which minimizes Modified Information Criterion (MIC) proposed by Watanabe [6]. However, MIC is designed for the pruning of the multi layered perceptrons. And, there has yet to be reported that the appropriate scheduling of the reduction of τ. On the contrary, our approach is designed for the selection of the size of the lerning model, and we give a scheduling method in the following section.

3.4 Optimal Annealing

If we reduce τ too much at one time, it is possible that the coefficient value w_i which should converges to 0 does not converge. Because the gradient of $g(x; \tau)$ in (11) has maximum value at $|x| = \tau$, we set the temperature parameter τ to the absolute value of the coefficient value w_i which has yet to be identified as non-convergeable parameter.

We use the following annealing algorithm.

1. $\tau \leftarrow \tau_{\max}$.
2. Corrected error backpropagation under τ until it converges.
3. If $\tau > \tau_{\min}$ then

$$\tau \leftarrow \max\left\{ \max_{|w_i| < \rho\tau} |w_i|, \tau_{\min} \right\}$$

and go to 2.

Here, ρ is a threshold value for determining whether w_i, which defines the temporal τ, is non-convergeable or not. The initial temperature $\tau_{\max}$ should be sufficiently large and the terminal temperature $\tau_{\min}$ should be sufficiently small such as $\tau_{\max} = 1.0 \times 10^{8}$ and $\tau_{\min} = 1.0 \times 10^{-8}$. We call this algorithm *Annealing Corrected error BackPropagation* (ACBP).

Table 1. Variables of the Boston house price estimation problem

$x_1 = 1$
$x_2 =$ per capita crime rate by town
$x_3 =$ proportion of residential land zoned for lots over $25,000$ sq.ft.
$x_4 =$ proportion of non-retail business acres per town
$x_5 =$ Charies River dummy variable (if tract bounds river then 1 else 0)
$x_6 =$ squared nitric oxides concentration (parts per 10 million)
$x_7 =$ squared average number of rooms per dwelling
$x_8 =$ proportion of owner-occupied units built prior to 1940
$x_9 =$ weighted distances to five Boston employment centres
$x_{10} =$ index of accessibility to radial highways
$x_{11} =$ full-value property-tax rate per \$10,000
$x_{12} =$ pupil-teacher ratio by town
$x_{13} = 1000(a - 0.63)^2$ where a is the proportion of African-American
$x_{14} =$ proportion of lower status of the population
$x_{15} =$ log Median value of owner-occupied homes in \$1000's

Table 2. Transition of the coefficients updated by ACBP. Bold face stands for the absolute value of the corresponding coefficient is smaller than 1.0E-8.

		w_3	w_4	w_8	w_{10}	w_{11}
ML	-	1.15E-03	2.75E-03	-4.12E-05	1.35E-02	-6.14E-04
$t = 1$	$\tau = 1.00$E8	1.15E-03	2.75E-03	-4.12E-05	1.35E-02	-6.14E-04
$t = 50$	$\tau = 2.75$E-3	1.07E-03	1.78E-03	-6.90E-06	1.32E-02	-5.81E-04
$t = 87$	$\tau = 1.07$E-3	9.05E-04	5.28E-04	-8.17E-06	1.26E-02	-5.28E-04
$t = 122$	$\tau = 5.28$E-4	8.22E-04	1.29E-04	-7.99E-06	1.23E-02	-5.05E-04
$t = 144$	$\tau = 4.04$E-4	9.19E-04	7.49E-05	8.11E-07	1.23E-02	-5.02E-04
$t = 181$	$\tau = 7.49$E-5	1.02E-03	2.97E-06	1.49E-06	1.28E-02	-5.28E-04
$t = 186$	$\tau = 2.97$E-6	1.02E-03	**7.90E-09**	-1.75E-08	1.28E-02	-5.27E-04
$t = 190$	$\tau = 1.75$E-8	1.02E-03	**-3.80E-11**	**2.24E-11**	1.28E-02	-5.27E-04
$t = 192$	$\tau = 1.00$E-8	1.02E-03	**-5.72E-12**	**3.42E-12**	1.28E-02	-5.27E-04

4 Computer Simulations

4.1 Boston House Price Estimation with Linear Regression Model

We apply ACBP to a linear regression model on the Boston house price estimation problem [7]. A corrected version of the data, which is available from StatLib Datasets Archive [8], is used in this section. We convert some variables from the original ones. Table. 1 shows variables. We want to estimate x_{15} using $x_1 \sim x_{14}$. When we calculate AIC of all 2^{14} combinations of the variables, the minimizer has 12 variables excluding x_4 and x_8.

The question is whether the optimal combination can be obtained by ACBP or not. In our simulation with the golden section search in each step of quasi Newton method ($\rho = 0.9$, $\tau_{\max} = 1.0 \times 10^8$ and $\tau_{\min} = 1.0 \times 10^{-8}$), ACBP successfully converges to the optimal combination as shown in Table. 2. The number of total steps is 192 and the number of updating the temperature is 19.

Table 3. Average and standard deviation of the normalized mean squared test error of 50 trials. The best result and comparable methods according to the T-test at the significance level 5% are written in bold face.

data	BP	ACBP ($A = 1$)	ACBP ($A = \log N/2$)
kin-8fm	**1.00** × **0.37**	**0.96** × **0.32**	**0.99** × **0.36**
kin-8fh	1.00 × 0.94	**0.47** × **0.22**	**0.43** × **0.13**
kin-8nm	1.00 × 0.30	**0.88** × **0.22**	**0.92** × **0.19**
kin-8nh	1.00 × 0.18	**0.93** × **0.14**	**0.92** × **0.15**
kin-32fm	1.00 × 0.50	0.33 × 0.12	**0.24** × **0.02**
kin-32fh	1.00 × 0.86	0.35 × 0.57	**0.21** × **0.10**
kin-32nm	1.00 × 0.12	0.70 × 0.08	**0.57** × **0.03**
kin-32nh	1.00 × 0.17	0.69 × 0.08	**0.57** × **0.04**
pumadyn-8fm	**1.00** × **0.25**	**0.94** × **0.16**	**0.94** × **0.15**
pumadyn-8fh	1.00 × 0.22	0.82 × 0.11	**0.73** × **0.08**
pumadyn-8nm	**1.00** × **0.35**	**1.04** × **0.31**	**0.97** × **0.27**
pumadyn-8nh	1.00 × 0.19	0.84 × 0.10	**0.76** × **0.09**
pumadyn-32fm	1.00 × 0.18	0.66 × 0.05	**0.55** × **0.05**
pumadyn-32fh	1.00 × 0.15	0.69 × 0.07	**0.58** × **0.06**
pumadyn-32nm	1.00 × 0.16	0.77 × 0.07	**0.66** × **0.05**
pumadyn-32nh	1.00 × 0.11	0.76 × 0.07	**0.65** × **0.06**

Table. 2 shows only 5 coefficients and 9 temperatures when a certain amount of changes occurred.

4.2 Delve Datasets with Multi Layered Perceptron

We apply BP (error BackPropagation) and ACBP (Annealing Corrected error BackPropagation) to multi layered perceptrons (MLPs) on kin-family dataset and pumadyn-family dataset in DELVE datasets [9].

It is known that the bias of the log-empirical likelihood of a MLP is bigger than that of a regular statistical model [10]. Asymptotic order of the lower bound of the bias has been proved to be $O(\log N)$ [11,12,13]. Therefore we should set A in (8) to be larger than 1. In this paper, we tried $A = 1$ and $A = \log N/2$.

Table. 3 shows the average mean squared test error of 50 trials ($N = 128$, $H = 8$). Although the optimal A is still open to discuss, ACBP with $A = \log N/2$ gives the best result or comparable to the best one in all cases.

5 Conclusion

In this paper, we proposed a learning method called *Corrected Error Backpropagation* which maximizes the corrected log-likelihood which works like Akaike Information Criterion. This paper also shows an optimal scheduling of the temperature parameter of the approximated corrected log-likelihood. Applying to our method to a linear regression model on the Boston house price estimation problem and multi layered perceptrons on the DELVE datasets, the results showed the effectiveness of our method.

References

1. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: 8, Learning internal representations by error propagation. In: Parallel distributed processing, vol. 1, pp. 318–362. MIT Press, Cambridge (1986)
2. Fukumizu, K.: Dynamics of batch learning in multilayer neural networks. In: Proc. of 8th International Conference on Artificial Neural Networks (ICANN 1998), pp. 189–194 (1998)
3. Fukumizu, K.: Generalization error of linear neural networks in unidentifiable cases. In: Watanabe, O., Yokomori, T. (eds.) ALT 1999. LNCS, vol. 1720, pp. 51–62. Springer, Heidelberg (1999)
4. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
5. Akaike, H.: A new look at the statistical model identification. IEEE Transactions on Automatic Control 19(6), 716–723 (1974)
6. Watanabe, S.: A modified information criterion for automatic model and parameter selection in neural network learning. IEICE transactions on information and systems 78(4), 490–499 (1995)
7. Harrison, D., Rubinfeld, D.L.: Hedonic housing prices and the demand for clean air. Journal of Environmental Economics and Management 5(1), 81–102 (1978)
8. Vlachos, P.: StatLib Datasets Archive (2001), `http://lib.stat.cmu.edu/datasets`
9. Rasmussen, C.E., Neal, R.M., Hinton, G.E., van Camp, D., Revow, M., Ghahramani, Z., Kustra, R., Tibshirani, R.: The DELVE manual (1996), `http://www.cs.toronto.edu/~delve/`
10. Hagiwara, K., Toda, N., Usui, S.: On the problem of applying aic to determine the structure of a layered feedforward neural network. In: Proc. of International Joint Conference on Neural Networks (IJCNN 1993), vol. 3, pp. 2263–2266 (1993)
11. Hagiwara, K., Hayasaka, T., Toda, N., Usui, S., Kuno, K.: Upper bound of the expected training error of neural network regression for a Gaussian noise sequence. Neural Networks 14(10), 1419–1429 (2001)
12. Hagiwara, K.: On the problem in model selection of neural network regression in overrealizable scenario. Neural Computation 14(8), 1979–2002 (2002)
13. Fukumizu, K.: Likelihood ratio of unidentifiable models and multilayer neural networks. The Annals of Statistics 31(3), 833–851 (2003)

Self-Referential Event Lists for Self-Organizing Modular Reinforcement Learning

Johane Takeuchi, Osamu Shouno, and Hiroshi Tsujino

Honda Research Institute Japan Co., Ltd.
8-1 Honcho, Wako-shi, Saitama 351-0188, Japan
{johane.takeuchi,shouno,tsujino}@jp.honda-ri.com

Abstract. Hybrid systems consisting of model-based and model-free systems will be engaged in the behavior/dialog control systems of future robots/agents to satisfy several user's requirements and simultaneously cope with diverse and unexpected situations. We have constructed a modular neural network model based on reinforcement learning for model-free learning. For an effective hybrid system, the model-free learning system should be aware of the current targets. This can be achieved by automatically acquiring a list of important sequential events. We propose a basic mechanism that can automatically acquire the list of sequential events with confidence measures reflecting current situations.

1 Introduction

Future symbiotic robots/agents are expected to support humans in houses, offices, and so on. They will autonomously perform situation associated behaviors/dialogs and carry out tasks for users in the real world. The behavior/dialog control system of future robots/agents will be a model-based system enabling the products to work incipiently. Every users' situations are often both outside of our expectations and volatile. Model-based systems will likely fail to perform situation specific behaviors because they are not inherently attuned to every user's situations. These limitations will make robots/agents commercially uncompetitive, and moreover, not truly symbiotic. A possible option for overcoming these shortcomings is to introduce an on-line learning system that bonds situations with behaviors. We are striving to create an on-line learning system that is capable of working together with model-based systems.

The on-line learning system should function using only its experiences outside of pre-designed environmental models in order to complement model-based systems. Reinforcement learning (RL) [1] is a learning algorithm that makes a map between observed states and actions, which is very close to our requirements. The standard implementations of RL, however, require formulating the learning targets. In general, RL requires task-specific reward functions that are carefully tailored by designers. One has to adjust the system to make them function, so RL implicitly depends on environmental models. In the standard implementations of RL, value functions are represented by means of tables. These tables are usually also created by designers. To reduce the dependence on hand-wired designs, we

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 228–235, 2009.

constructed an RL system capable of functioning in several kinds of state transitions without hand-wired adjustments for each transition. This is crucial in order to achieve learning systems which can cope with real situations where several kinds of state transitions potentially emerge. In our previous model [2], we introduced internal rewards to reduce hand-wired factors of reward functions. In addition, the system consisted of neural networks equipped with a modular reinforcement learning algorithm. We found that the combined system with both modular networks and the internal reward generators led to performance that converged to the optimal sequences of actions in multiple tested transitions without any specific adjustments for each transition. The tested transitions included not only a Markov decision process (MDP) but also a transition under partially observable conditions. We also demonstrated that the modular neural network model achieved comparable performance to one implemented with a table representation [3]. We call the constructed on-line learning model SOMRL (Self-Organizing Modular RL).

Even if SOMRL could learn several kinds of unknown transitions, it is insufficient for constructing a hybrid system for robot controls. The learning system should suggest selected actions with ‘sufficient reasons’ because the robots/agents need to determine their actions depending not only on current states but also on current targets. For example, a temporal target of the robot can be different from the optimized actions for RL in actual situations because the robot should deal with multiple targets to satisfy several user’s requirements. If the learning system has the information about current targets, it can consider whether actions suggested by on-line learning are effective for current targets. In principle, however, RL cannot even distinguish between several goals. By means of recorded sequence lists during learning, we may be able to identify target goals for each selected action. It is not trivial to make the lists that are easy to use for providing target information because the raw transitions usually contain errors and probabilistic factors. Moreover, taking an action at a state usually ends in several targets. We require confidence measures to report the current target of the on-line learning system. We named the acquired events lists SREL (Self-Referential Event Lists), they were formulated so that the learning system could report itself, that is, what it is targeting now. In this study, we propose a basic mechanism that can automatically acquire sequential events with confidence measures in unknown, probabilistic, fluctuating, partially-observable and open-ended situations.

2 Background

In this study, we referred to intrinsically motivated reinforcement learning (IMRL) studies [5,6] to construct the algorithm. IMRL studies represented acquired skills of agents by means of options [7]. Options are the minimal extension of actions. IMRL studies also suggested a method to acquire options by introducing intrinsic motivations where specified salient events were temporal motivations to acquire new options and modify them. The formulations of option reinforcement learning depended on table representations where any state information can be available

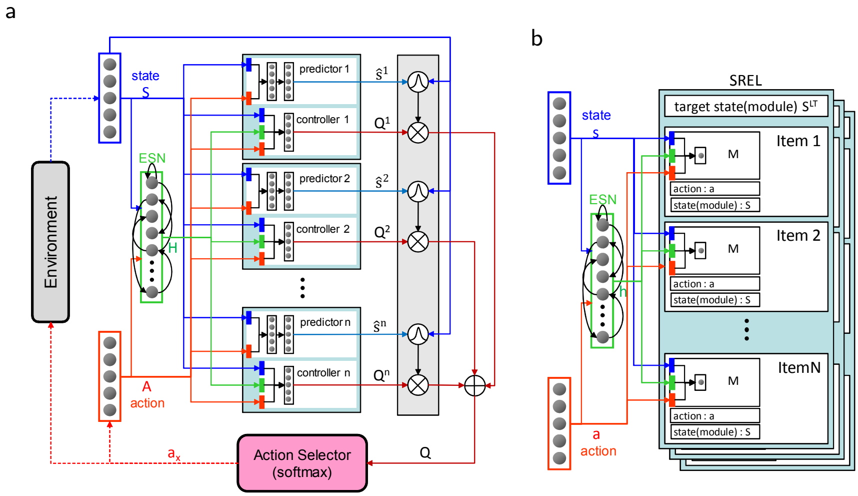


Fig. 1. a) A schematic diagram of the SOMRL. The predictor network in each module predicts the current state $\widehat{s_t}^n$ from the previous observed state s_{t-1} and the previous action a_{t-1}. Each predictor network contains one hidden layer. The outputs of the controller layer in each module (Q^n) are Q-function vector values for each action. The inputs of each controller layer are the observed state s_t (s), the previous action a_{t-1} (a), and the history information h_{t-1} (h). The history information is represented with an echo state network (ESN) reservoir [4]. Each state was represented by a fifteen-dimensional normalized vector consisting of random values. The previous actions were represented by ten-dimensional unit vectors. For the controller in each module, we adopted a gradient-descent SARSA(λ) algorithm [1]. b) A schematic diagram of the SREL.

for calculations at any time so that they can perform a backup of all possibilities for option policy learning. These requirements can hardly be fulfilled in neural network representations.

We were also affected by the idea of temporal-difference networks (TD networks) [8]. There is a proposed method for on-line discovery of TD networks [9]. But the method to calculate predictions for TD networks is difficult to apply for probabilistic transitions without pre-designed environmental models.

We formulate SREL for SOMRL [2,3]. One of the advantages in using neural networks is that we can reduce the dependencies on input space dimensions. Taking advantage of this property, we could make reinforcement learning function in several kinds of transitions including a simple MDP and partially observable conditions without properly adjusting input information. Table representations directly depend on the input dimensions so it is necessary to adjust input information for each transition. The learning speed of a single neural network is, however, generally much slower than that of table representations especially when a single neural network tries to learn all sequences. Hence we introduced a modular structure to decompose the sequences into several modules. Figure 1a shows a schematic diagram of SOMRL described in our previous article [3]. We previously applied [2] the modular reinforcement learning algorithm suggested by Doya, *et al.* [10], which was effective in on-line learning. It was composed of n modules, each of which consisted of a state prediction model and an RL controller. The action output of the RL controllers, as well as the learning rates of both the predictors and the controllers, were weighted by the predictability levels, which is a Gaussian softmax function of the errors in the outputs of the predictions [10]. The most recent SOMRL was based on modular RL algorithm suggested by Nishida *et al.* using the modular network self-organization map (mnSOM) [11] that is an extension of the conventional Kohonen's SOM [12]. In the mnSOM each SOM node extended to a functional module such as the module in our model, in which the winning node was the module that has the lowest prediction error at a time. By combining the mnSOM with the parameter-less SOM proposed in [13], we made the on-line version of mnSOM. (We did not adopt an original SOM because it was empirically too sensitive to the actual number of input states in our model.)

3 Self-Referential Event List

Each SREL is compromised of one target state s^{LT} and a list $\mathcal{I}^L$ and each item of the list contains a pair of an action (a_j) and a state (s_i) and a confidence measure value $\mathcal{M}^L$ for this pair (Figure 1b). We expected that the agent would earn rewards by transiting to the target states of each SREL. Each state was not predefined in the SOMRL model because our model-free learning should function in several kinds of transitions without hand adjustments. In stead of using defined states, each state can be represented by the winning module index S_i in the SOMRL model. We should note that each module may represent multiple actual states. Hence, we utilized almost the same neural network as the controllers in SOMRL. Each item in SREL has a two-layered neural network to represent a confidence measure value $\mathcal{M}^L$ so that the confidence measures can reflect the actual inputs. The input sources of the network are the same as the controllers in SOMRL, current observed states, previous actions and history. Each target state of SREL also points to a specific module S^{LT} in SOMRL, which could cause errors when the targeted module represents plural states. We could correct these errors by distinguishing each represented state via another classification algorithm such as the nearest neighbor method. In this study, however, we evaluated the system by leaving them as errors for simplifications. We formulated the learning algorithm to approximate each confidence measure value in SREL.

$$\mathcal{M}^L(S, a) = E\{\mathtt{I}(S_{t+k})|S, a, S^{LT}\},$$

where $t + k$ is a time when a state is outside of $\mathcal{I}^L$ and E means expectation. $\mathtt{I}$ is,

$$\mathtt{I}(S_t) = \begin{cases} 1 & \text{if } S_t = S^{LT} \\ 0 & \text{else.} \end{cases}$$

We can approximate $\mathcal{M}^L(S, a)$ via on-line TD-learning. If $S_{t+1} = S^{LT}$, then,

$$\mathcal{M}^L(S_t, a_t) \leftarrow \mathcal{M}^L(S_t, a_t) + \alpha[1 - \mathcal{M}^L(S_t, a_t)], \tag{1}$$

where α is an update state constant. If $S_{t+1} \neq S^{LT} \wedge \{S_{t+1}, a_{t+1}\} \in \mathcal{I}^L$, then,

$$\mathcal{M}^L(S_t, a_t) \leftarrow \mathcal{M}^L(S_t, a_t) + \alpha[\mathcal{M}^L(S_{t+1}, a_{t+1}) - \mathcal{M}^L(S_t, a_t)]. \tag{2}$$

In the other cases,

$$\mathcal{M}^L(S_t, a_t) \leftarrow \mathcal{M}^L(S_t, a_t) + \alpha[0 - \mathcal{M}^L(S_t, a_t)]. \tag{3}$$

This learning takes place in all the SRELs that satisfy $\{S_t, a_t\} \in \mathcal{I}^L$.

Barto *et al.* [5,6] also suggested a method to generate options. We based on their algorithm for automatically producing SRELs. When the agent earns a positive reward r_t at $S = S_t$, the system searches the already acquired SRELs that contains the target state as $S_t = S^{LT}$. If any SRELs did not contain the target state S_t, the system generates a SREL targeted S_t and adds $\{S_{t-1}, a_{t-1}\}$ to $\mathcal{I}^L$. If the system finds a SREL of which S^{LT} is S_t and $\mathcal{I}^L$ does not contain $\{S_{t-1}, a_{t-1}\}$, it added the pair item to the list. When the rewards are zeros, each SREL tries to grow the list $\mathcal{I}^L$. If a SREL has maximum $\mathcal{M}^L(S_t, a_t)$ at time t, the pair $\{S_{t-1}, a_{t-1}\}$ is added to the SREL except in the case where it already contains the pair or the action-value (Q) for $\{S_t, a_t\}$ of SOMRL is not positive.

4 Evaluation

This section describes the evaluations of our proposed model using an environment in which the kind of transition was abruptly changed from an MDP to a partially observable conditioned transition. Figure 2a shows the MDP transition. At initial state s_0, if the agent selected actions sequentially in the order of the subscriptions, external rewards were earned, otherwise, the agent received no positive external reward. We engaged in continuing tasks so the transition repeated from the initial state even after earning an external reward. The probability of transition from s_0 to s_1 with action selection a_0 is 0.3. Other transitions of the correct sequence to earn external rewards were 0.9. In addition, the observed states were probabilistically changed. For example, in state s_0, the agent could observe a state signal o_{00} or o_{01} with a probability of 0.5. Each state s_n was represented by o_{n0} and o_{n1}, so the number of observed state signals was twelve in this environment. All state signals o_{nx} were fifteen dimensional vectors that were randomly generated; therefore, o_{n0} and o_{n1} for the same s_n were not correlated except coincidentally. We also let in the partially observable transition depicted in Fig. 2b. In this case, if the agent succeeded in earning an external reward, the unobservable state of the environment changed to another state. Thus, because of the unobservable state, the state transitions were changed and the agent had to select another sequence of actions to earn another external reward. Each trial ended when the agent received two external rewards. The agent had to pass two

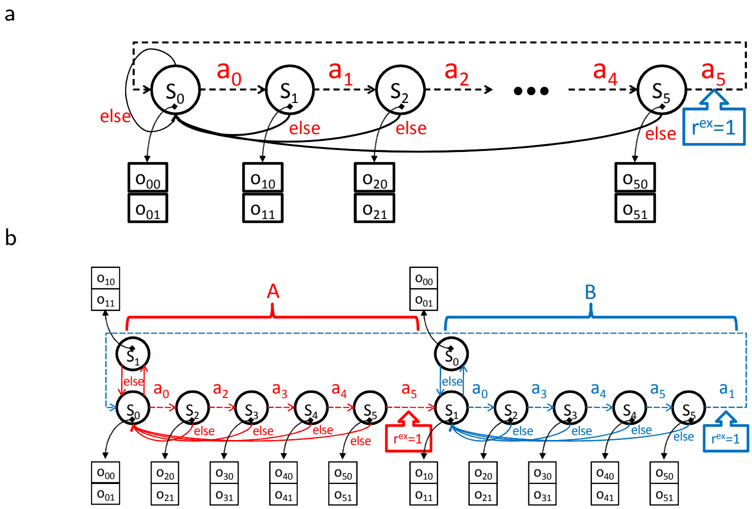


Fig. 2. a) Test continuing MDP transition. An external reward $r^{ex} = 1$ is earned after sequential events. Dashed lines indicate probabilistic transitions. b) Test continuing partially observable conditioned transition. The optimum action selection when the agent starts from the state s_0 is $a_0 \rightarrow a_2 \rightarrow a_3 \rightarrow a_4 \rightarrow a_5$ (A), thereby receiving r^{ex} and then $a_0 \rightarrow a_3 \rightarrow a_4 \rightarrow a_5 \rightarrow a_1$ (B) whereby the agent receives another r^{ex}.

probabilistic *alternating* transitions of $p = 0.3$ to earn two external rewards. The memory information to solve the partially observability was previous actions at transitions from s_3 to s_5. When the agent was in state s_2, the agent could not distinguish the hidden states from the previous action because the previous actions were the same in both hidden states. History information was necessary to distinguish the situation. We refer to this transition as HOMDP (High Order MDP).

Figure 3a shows learning curves for the average performance of 1000 trial sets (each set consisting of 2000 trials). Each trial set was simulated with different state vector values. Each trial started with the final states of the immediately preceding trials, and stopped either when the agent was rewarded, or when time ran out. Every transition in all cases cost negative rewards -0.05 in SOMRL where the number of modules was 25. We set $\alpha = 0.1$ in equations (1)-(3). In this environment, the transition was abruptly changed from MDP to HOMDP at 1000 trials. Nothing about this change was notified to the learning system. We investigated the averaged concordance rate of acquired lists of state-action pairs in all SRELs with the correct ones (Fig. 3b). The correct pairs were generated by adding the correct sequences whenever they actually occurred during the simulation. The correct sequence means that the agent chose correct actions to earn external rewards and the state transition progressed toward earning rewards. For example, in the MDP, when the agent selected action a_1 at state s_1 and the state was transit to s_2, this sequence was a correct one. Both the SREL and the correct pair list were never reduced even after the transition change

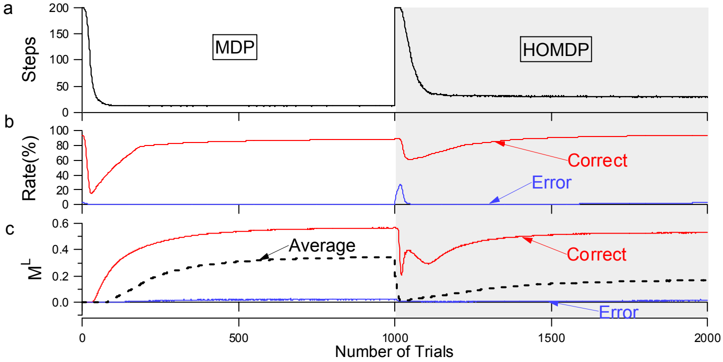


Fig. 3. a) The learning curves SOMRL in the environment. The ordinates indicate the number of steps to earn external rewards before time runs out (200 steps). The abscissas indicate the number of trials the agent had attempted in each set. b) The averaged concordance rate of acquired state-action paris in SRELs. 'Correct' means (the number of acquired pairs that concord with the correct ones)/(the number of all correct pairs). 'Error' is (the number of acquired pairs that are not matched with any correct ones)/(the number of all acquired pairs). c) The averaged maximum confidence measures within the 'Correct' pairs and the 'Error' pairs. 'Average' denotes the averaged values of all confidence measures.

Table 1.

	1	2	3	4	5
s	o_{11}	o_{20}	o_{31}	o_{41}	o_{51}
a	a_0	a_3	a_4	a_5	a_1
$\mathcal{M}^L$	0.11	0.36	0.44	0.51	0.59

from MDP to HOMDP. The averaged error rates in Fig. 3b were nearly zero. In contrast, the correct ones reached over 90 %. Figure 3c shows the averaged maximum confidence measures within the correct pairs and the erroneous pairs. The averaged maximum confidence measures reached around 0.5: that was correct because the target states were separated by probabilistic changing state signals. For example in the MDP transition, the target could be o_{00} or o_{01}. These confidence measures indicate the beliefs to reach each specific target. Table 1 shows a SREL of a final sequence that was HOMDP started from o_{11} (s_1) and ended at o_{01} (s_0). The confidence measures $\mathcal{M}^L$ roughly indicated the probabilistic property of this transition where the probabilities were 0.9 from s_2 to s_5 and the transition from s_1 to s_2 was 0.3.

5 Discussion

We applied on-line TD-learning to calculate confidence measures of acquired sequential event lists. These acquired confidence measures reflected the recent policy of reinforcement learning, environmental changes and fluctuations. They indicate the extent to which each state-action pair was valid to achieve each target at any time. In order to grow each list, we referred to confidence measures and action values (Q) of SOMRL because we thought that it was not necessary to add events of low confidence and small action values. We confirmed the low rate to acquire inaccurate sequences by the result. By using SRELs, the learning system is able to suggest an action with not only expectation value of rewards but also target states with confidence measures. We believe that these mechanisms can bridge between model-free learning and model-based systems.

References

1. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
2. Takeuchi, J., Shouno, O., Tsujino, H.: Modular neural networks for reinforcement learning with temporal intrinsic rewards. In: Proceedings of 2007 International Joint Conference on Neural Networks, IJCNN (2007) (CD-ROM)
3. Takeuchi, J., Shouno, O., Tsujino, H.: Modular neural networks for model-free behavioral learning. In: Proceedings of the 18th International Conference on Artificial Neural Networks (ICANN), vol. I, pp. 730–739 (2008)
4. Jaeger, H.: The 'echo state' approach to analysing and training recurrent neural networks. Gmd report 148, German National Research Center for Information Technology (2001)
5. Barto, A.G., Singh, S., Chentanez, N.: Intrinsically motivated learning of hierarchical collection of skills. In: Proceedings of the 3rd International Conference on Developmental Learning, ICDL (2004)
6. Singh, S., Barto, A.G., Chentanez, N.: Intrinsically motivated reinforcement learning. In: Advances in Neural Information Processing Systems 17, pp. 1281–1288. MIT Press, Cambridge (2005)
7. Sutton, R.S., Precup, D., Singh, S.P.: Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence 112(1-2), 181–211 (1999)
8. Sutton, R.S., Tanner, B.: Temporal-difference networks. In: Advances in Neural Information Processing Systems 17 (NIPS 2004), pp. 1377–1384 (2005)
9. Makino, T., Takagi, T.: On-line discovery of temporal-difference networks. In: Proceedings of the twenty-fifth international conference on machine learning, ICML (2008)
10. Doya, K., Samejima, K., Katagiri, K.i., Kawato, M.: Multiple model-based reinforcement learning. Neural Computation 14, 1347–1369 (2002)
11. Nishida, S., Ishii, K., Furukawa, T.: An online adaptation control system using mnSOM. In: King, I., Wang, J., Chan, L.-W., Wang, D. (eds.) ICONIP 2006. LNCS, vol. 4232, pp. 935–942. Springer, Heidelberg (2006)
12. Kohonen, T.: Self-Organizing Maps. Springer, Heidelberg (1995)
13. Berglund, E., Sitte, J.: The parameterless self-organizing map algorithm. IEEE transactions on neural networks 17(2), 305–316 (2006)

Generalisation Performance vs. Architecture Variations in Constructive Cascade Networks

Suisin Khoo and Tom Gedeon

School of Computer Science
College of Engineering and Computer Science
Australian National University
ACT 0200 Australia
{suisin.khoo,tom}@cs.anu.edu.au

Abstract. Constructive cascade algorithms are powerful methods for training feedforward neural networks with automation of the task of specifying the size and topology of network to use. A series of empirical studies were performed to examine the effect of imposing constraints on constructive cascade neural network architectures. Building *a priori* knowledge of the task into the network gives better generalisation performance. We introduce our Local Feature Constructive Cascade (LoCC) and Symmetry Local Feature Constructive Cascade (SymLoCC) algorithms, and show them to have good generalisation and network construction properties on face recognition tasks.

1 Introduction

The functionality and complexity of a backpropagation trained neural network (NN) are greatly influenced by the network architecture, number of neurons, and neuron connectivity. The greater number of connections and the larger the possible magnitude of the weights, the better the NN is able to model complex functions.

A constructive cascade neural network is a feedforward neural network in which the network architecture is built during the learning process, in order to obtain a good match between network complexity and the complexity of the problem to solve [1].

In some highly regular tasks such as image recognition, a priori knowledge about the task can be built into the network for better generalisation performance. Classical works in visual pattern recognition have shown the advantages of using local features and combining them to form higher order features. Extracting local features can be viewed as a way of reducing the space of possible functions that can be generated without overly reducing the network's computational power [2].

In this paper, we propose two constructive cascade algorithms that incorporate a priori knowledge about the problem to be solved, i.e. face recognition, into the design of the architecture and explore the relationship between the generalisation performance and variations on the architecture. These are the Local Feature Constructive Cascade (LoCC) and Symmetry Local Feature Constructive Cascade (SymLoCC) algorithms. They have lesser number of free parameters but preserve the good generalisation properties of constructive cascade algorithms. Unlike other face recognition technique such as in [3] our approach does not require normalisation and pre-processing of data sets or additional feature extraction procedures.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 236–243, 2009.

2 Method

2.1 Neural Network Topology

Constructive algorithms such as CasPer and Cascade Correlation (CasCor) start with the minimal size NN architecture often with no hidden neurons [4] as in Fig. 1. Initially, all input neurons are fully-connected to the output neurons. The number of connections in the initial network would be very large for face recognition, hence LoCC and SymLoCC start with an initial network with one hidden layer as shown on the left in Fig. 2.

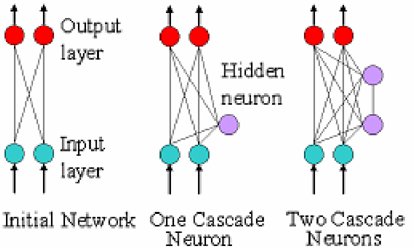


Fig. 1. Cascade Correlation

The initial architecture of our algorithm is a slight modification of the two layer 8x8 architecture proposed in [5] which was shown to give good generalisation performance on handwritten digit recognition and face recognition. Instead of a two-dimensional 16 by 16 hidden layer with a total of 256 hidden neurons, our algorithm uses a reduced number of hidden neurons, which is 8 by 8 hidden layer, to achieve smaller network size with similar good generalisation performance. Each hidden neuron functions as a local feature extractor that receives inputs from the corresponding receptive field in the input layer. The hidden neurons are thus partially connected to the input neurons but are fully-connected to the ten output neurons.

The middle diagram in Fig. 2 shows the network structure after one cascade layer is added. Instead of adding hidden neurons to the network one at a time during the learning process as in typical constructive cascade algorithms such as CasCor [6], our algorithm adds a number of cascade chunks (cascade layers), each of a fixed size set prior to training. Treadgold and Gedeon used a similar approach to build cascade towers for their algorithm [4]. In order to limit the functionality of the cascade layer and to reduce the overall number of hidden neurons, the size of each cascade layer is smaller than the hidden layer. Each cascade layer consists of 4 by 4 neurons, each extracts local features from both the input and the hidden layer in the initial network. Preserving the characteristic of constructive cascade algorithms, the cascade layers receive input from all preceding layers and are fully-connected to the output layer.

There are three variants of the two architectures presented in this paper, with the same overall structure shown in Fig. 2. They differ in the size and structure of the receptive fields. In the first variant, each neuron in the hidden layer takes its inputs from 25 neurons on the input layer situated in a 5 by 5 square neighbourhood. For neurons in the hidden layer that are one unit apart, their receptive fields in the input layer are four units apart. Hence, their receptive fields overlap by one row or column of units.

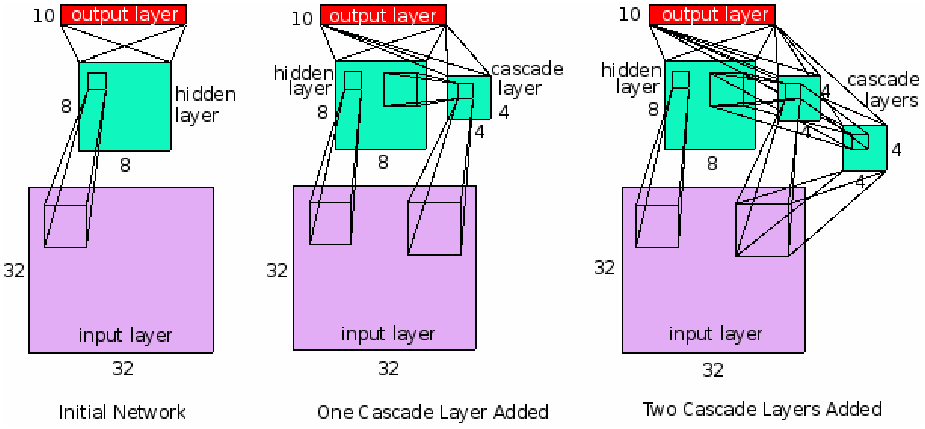


Fig. 2. Local Feature Constructive Cascade Neural Networks

In the first variant, each neuron in a cascade layer takes its inputs from the input layer situated in a 9 by 9 neighbourhood and from the hidden layer situated in a 3 by 3 neighbourhood. For each neuron that is adjacent, the input patches from the input layer are eight units apart and are two units apart on the hidden layer. In other words, 9 by 9 patches on the input plane have one row or column of unit overlapped and the 3 by 3 patch on the hidden layer has one row or column of unit overlapped. Each cascade layer has one-to-one connection to all preceding cascade layers.

In the second variant, each hidden neuron receives 36 inputs from a 6 by 6 square neighbourhood in the input layer with 2 rows or columns of units overlapped. In the cascade layer, each neuron receives inputs from a 10 by 10 patch with 2 rows or columns of neurons overlapped. Neurons in the cascade layer also receive inputs from 16 hidden neurons in the hidden layer, arranged in 4 by 4 receptive fields with two rows or columns of neurons overlapped. As in the first variant, all neurons in each cascade layer have one-to-one connections with all preceding cascade layers.

Finally, the third variant employs larger receptive fields than both previous architectures. Hidden neurons in the initial network each takes a 7 by 7 receptive field in the input layer. Neurons in cascade layers each takes a 11 by 11 receptive field in the input layer and from the 5 by 5 receptive field in the hidden layer. As before, each cascade layer has one-to-one connections with all preceding cascade layers.

2.2 Data Set

The two data sets used were originally proposed by Georghiades, Belhumeur, and Kriegman [7] and is available from the Yale Face Database B. The original data set contains a total of 5,850 single light source images of 10 subjects each seen under 576 viewing conditions (9 poses of 64 illumination conditions and 1 with ambient illumination). The images are 8-bit grey scale with size 480 height x 640 width. Due to its massive size, not all of the data set in the Yale Face Database B was employed in the first data set. Thirteen images of each subject under nine different poses were randomly selected from the original data set. Of these images, 630 of them are designated as the training set, and 360 as test set. Each face image was resized using nearest-neighbour interpolation [8] into a 24 x 32 image, with added 0s to form a 32 x 32 images. The second consists of 650 frontal view images in total, split into 450 training data and 200 testing data, randomly.

2.3 Training Methodology

Human faces possess some symmetric characteristics. SymLoCC implements this knowledge by adjusting all weights leaving the input layer so that the weights for neurons in the right-half of the square plane mirror the value of the weights in the left-half of the plane (i.e. the weights are shared). Hence, the number of free parameters from input layer to hidden and cascade layers are 50% less than in the non-symmetric algorithm LoCC.

Before training our neural networks, all weights and biases of the network are initialised using the Nguyen-Widrow method [9]. The activation function used in all networks is the hyperbolic tangent function. Hyperbolic tangent functions are symmetric functions, which are believed to be able to yield faster convergence than non-symmetric functions. Resilient propagation (RPROP) [10] was use for all networks learning. RPROP is an adaptive learning algorithm which performs a direct adaptation of the weight step size based on local gradient information. Instead of taking into account the magnitude of the error gradient as seen by a particular weight, RPROP uses only the sign of the gradient. This allows the algorithm to adapt the step size without the size of the gradient interfering with the adaptation process. Also, a weight biasing technique [11] is employed to bias the search direction of the RPROP algorithm in favour of the weights of the newly added neurons, setting different initial update values in the RPROP algorithm.

For each architecture variant, the network starts from the initial architecture and the learning process continues for a maximum of 100 epochs. A new cascade layer is added to the network when either the maximum epoch is reached or the MSE is less than or equal to 0.03. The input patterns were presented in a consistent order, batch trained. Each experiment was performed 5 times with different initial conditions. The performance function is the standard mean squared error (MSE), the output layer was composed of 10 units, one per class, and we used a winner-takes-all method to classify the networks' output error on the test data set.

3 Experimental Results

We have evaluated our method using two subsets of Yale Database B as described in Section 2.2. The problem to be solved is face recognition of ten subjects. Information of each face image is input into the network through 32 x 32 that is 1,024 input neurons. The 10 output neurons each represent one of the subjects. Table 1 shows the average of the best performance of the five repetitions of each experiment.

Addition of each cascade layer increases the number of weights by some 1,500 to 2,000. These increases lead usually to some increase in performance for each architecture-variant combination. The performance of most networks is quite good, ranging from a few results in the vicinity of 80%, but with more than half of the architecture-variants with results over 95%.The average of the best performance of the five

Table 1. Average of best performance on both data sets

Architecture			Number of weights	Generalisation Performance % (all poses)	Generalisation Performance % (frontal pose)
LoCC	551-991-331	initial network	2,314	94.2	95.7
		cascade layers: 1	3,930	95.2	96.5
		2	5,562	97.7	99.5
		3	7,210	98.6	99.0
	662-10102-442	initial network	3,018	94.5	93.5
		cascade layers: 1	5,050	94.8	95.0
		2	7,098	98.2	99.2
		3	9,162	97.7	99.1
	773-11113-553	initial network	3,850	92.2	92.9
		cascade layers: 1	6,362	94.7	95.8
		2	8,890	98.6	98.8
		3	11,434	97.6	98.6
SymLoCC	551-991-331	initial network	1,514	83.3	92.0
		cascade layers: 1	2,482	90.8	94.1
		2	3,466	97.9	99.1
		3	4,466	97.6	98.8
	662-10102-442	initial network	1,866	82.6	88.5
		cascade layers: 1	3,098	91.9	95.0
		2	4,346	97.8	98.4
		3	5,610	97.7	98.5
	773-11113-553	initial network	2,282	86.7	81.8
		cascade layers: 1	3,826	93.6	94.1
		2	5,386	97.8	97.6
		3	6,962	98.3	98.7

repetitions of each experiment performed using the first data set is plotted in Fig. 3, and similarly for the second data set in Fig. 4. The connecting lines in the graph show the growth of complexity of each architecture from the initial network to the final network structure with three cascade layers.

In Fig. 3 we can see a number of patterns. Most obviously, as the number of weights increases, there is generally an increase in the generalisation performance of the networks. The best results are those located in the top-left corner, hence the best result is either *LoCC-551-991-331 initial network*, or *SymLoCC-551-991-331 with two cascade layers*. The choice between them would be context dependent. I.e., is the 50% increase in number of weights worthwhile for 3.7% increase in performance? Alternatively stated, is this worthwhile for a $^{2}/_{3}$ decrease in the remaining error (from 5.8% to 2.1%)? In subsequent discussion, we will focus on differential effects of our various architectural variations, and the effect of sequential addition of cascade layers.

For the first of our architectures, the local feature constructive cascade (LoCC) network, shown in Fig. 3 by the solid lines, we can see that addition of a cascade layer increases the weights appreciably, while only slightly increasing the generalisation ability. This pattern holds for the 2nd and 3rd variants, as they also show little improvement in generalisation. The addition of a second cascade layer has a beneficial effect, in that there is a more significant increase in generalisation ability. The addition of a third cascade layer is slightly detrimental in two cases and mildly beneficial in one case, so this is not useful as we add weights without improving generalisation ability much or at all.

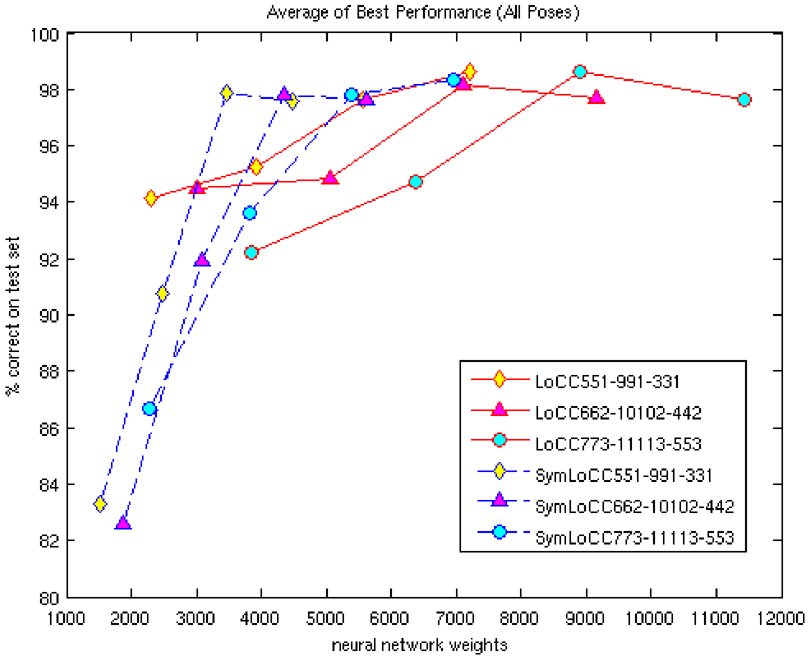


Fig. 3. LoCC and SymLoCC results using the first data set (all poses)

For the second architecture, using symmetry (SymLoCC), shown in Fig. 3 by dashed lines, the performance starts at much lower values. The addition of a cascade layer improves generalisation for all three variants. At this point the second architecture variants with a cascade layer have very much the same number of weights as the first architecture and no cascade, but with lower generalisation. On the addition of a second cascade layer, again we find an increase in performance for all three variants, and now we achieve a situation of lower weights but higher performance than the next step of the first architecture. A third cascade layer produces little effect.

We can note a few subtle observations. For the second architecture, as we move from the 1st variant to the 3rd, adding the first two cascade layers, the slope of improvement decreases from 1st variant to 3rd variant on the addition of the first cascade layer, and further decreases in the same way on the addition of the second cascade layer. A possible explanation is that the patch sizes are overall better in the 1st variant and worsen to the 3rd variant.

In Fig. 4, we have slightly better data, in that the frontal poses are by definition more symmetrical and might benefit more from the symmetry incorporated in our second architecture. No pre-processing has been done to align faces in the images.

Our first architecture (solid lines) has similar effects as in Fig. 3, in that the addition of the first cascade layer has some effect, which is increased by the second cascade layer. At this point the overall best result is reached, the 1st variant with two cascade layers has 99.5% accuracy on the test set. A third cascade layer is not beneficial.

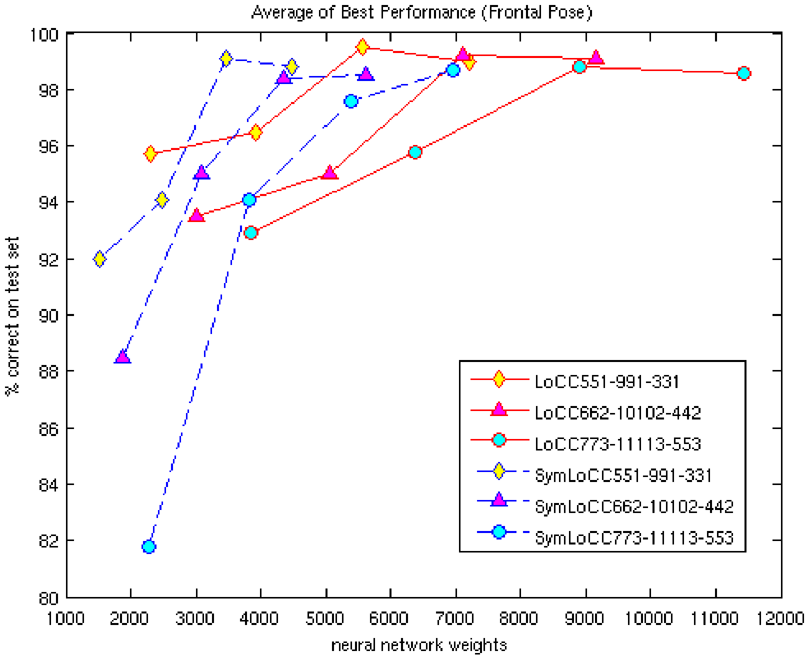


Fig. 4. LoCC and SymLoCC results using the second data set (frontal pose only)

Our second architecture (dashed lines) also has similar results to Fig. 3, in that results start lower than the first architecture, the addition of two cascade layers produces results which are substantially improved, and better than the first architecture.

For Fig. 4, we can make a different illustration of the tradeoff if we assume that absolute performance is most important. In that case the best networks have 3,466 weights 99.1%, and 5,562 weights and 99.5% generalisation performance, respectively. Depending on the measurement error, the percentage results may not be reliably distinguishable so clearly the version with significantly lower weights is best.

4 Conclusion and Future Work

We have introduced our algorithm for constructing cascade networks for face recognition using our notion of cascade layers, by constraining the cascade process by adding chunks of 4 x 4 neurons. We have examined a number of variants of this model, focusing on the sizes of patches taken by each layer from the preceding layer. We have extended this model by introducing a symmetry component. From our testing we have shown that our models work well on a standard face image database. We have demonstrated that restricting the data to just the frontal pose improves all our results. An alternative expression of this statement demonstrates the strength of our approach: if we take the frontal pose as the baseline, then using multiple different poses cost only a 0.9% drop in maximum performance.

Our future work will be to further examine the architecture variations in our model. The three variations discussed here increased all the patch sizes from 1st variant to 2nd, and from 2nd variant to 3rd variant, however the patch sizes could be varied independently. We will also investigate the effects of aligning the frontal pose images on the images.

References

1. Kwok, T.-Y., Yeung, D.-Y.: Constructive Algorithms for Structure Learning in Feedforward Neural Networks for Regression Problems. IEEE Trans. on Neural Networks 8, 630–645 (1997)
2. LeCun, Y.: Generalization and Network Design Strategies. Technical report, Dept. of Computer Science, University of Toronto (1989)
3. Grudin, M.A.: On Internal Representations in Face Recognition Systems. Pattern Recognition 33, 1161–1177 (2000)
4. Treadgold, N.K., Gedeon, T.D.: Exploring Architecture Variations in Constructive Cascade Networks. In: IEEE Int. Jt. Conf. on Neural Networks, pp. 343–348 (1998)
5. Khoo, S.: Application of Shared Weight Neural Networks in Image Classification. Honours Thesis, Dept. Computer Science, Australian National University (2008)
6. Fahlman, S., Liebiere, C.: The Cascade-Correlation Learning Architecture. Technical report, School of Computer Science, Carnegie Mellon University (1990)
7. Georghiades, A.S., Belhumeur, P.N., Kriegman, D.J.: From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose. IEEE Trans. on Pattern Analysis and Machine Intelligence 23, 643–660 (2001)
8. Cover, T.M., Hart, P.E.: Nearest Neighbor Pattern Classification. IEEE Trans. on Information Theory IT-13, 21–27 (1967)
9. Nguyen, D., Widrow, B.: Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights. In: IEEE Int. Jt. Conf. on Neural Networks, pp. 21–26 (1990)
10. Riedmiller, M., Braun, H.: A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In: IEEE Int. Conf. on Neural Networks, pp. 586–591 (1993)
11. Treadgold, N.K., Gedeon, T.D.: Increased Generalization through Selective Decay in a Constructive Cascade Network. In: Proc. 1998 IEEE Int. Conf. on Systems, Man, and Cybernetics, pp. 4465–4469 (1998)

Synchronized Oriented Mutations Algorithm for Training Neural Controllers

Vincent Berenz and Kenji Suzuki

Artificial Intelligence Laboratory
Department of Intelligent Interaction Technologies
Graduate School of Systems and Information Engineering
University of Tsukuba, Japan
http://www.ai.iit.tsukuba.ac.jp

Abstract. Developing neural controllers for autonomous robotics is a tedious task as the desired state trajectory of the robot is very often not known in advance. This led to the large success of evolutionary algorithm in this field. In this paper we introduce SOMA (Synchronized Oriented Mutations Algorithm), which presents an alternative for rapidly minimizing the parameters characterizing a given individual. SOMA is characterized by its easy implementation and its flexibility: it can use any continuous fitness function and be applied to optimize neural network of diverse topologies using any kind of activation functions. Contrary to evolutionary approach, it is applied on a single individual rather than on a population. Because the procedure is very fast, it allows for rapid screening and selection of good candidates. In this paper, the efficiency of SOMA at training ordered connection feed forward networks on function modeling problem, classification problem and robotic controllers is investigated.

1 Introduction

The field of autonomous robotic requires the development of automatic methods for controller synthesis that do not need hand coding or in depth human knowledge of the robot's task. Artificial neural networks (ANN) have been used successfully as such robot controllers, but pose the problem of optimizing ANN structure in the context of sparse, noisy and delayed rewards. For example technics such as back-propagation require a desired state trajectory for comparison with the actual output trajectory of the network so that an error signal can be computed. In autonomous robotics, no such desired state trajectory generally exists. This is why evolutionary algorithm are so broadly used: contrary to gradient descent based algorithm, they do not depend on gradient information and thus are quite suitable for problems where such information is unavailable or very costly to obtain or estimate. They can even deal with problems where no explicit and/or exact objective function is available [1].

The present paper presents the Synchronized Oriented Mutation Algorithm (SOMA), an alternative or a completion to evolutionary strategies for training

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 244–251, 2009.

ANN. SOMA is characterized by a very simple implementation, can be used with any continuous fitness function (does not require learning/validation sets based fitness function), does not rely on the derivative of neuron's activation function, can be used with diverses ANN topologies (not restricted to layered feed forward networks) and can be used to update any kinds of free parameters (not restricted to connection's weights - useful if the activation function contains several parameters to be optimized or for avoiding the use of bias neuron).

After presenting the algorithm itself, we present the results obtained using it for: minimizing a three variables equation and training fully connected feed forward networks (modeling trigonometric/polynomial function, solving a classic classification problem and training robotic controller)[1].

2 General Principle

2.1 Oriented Mutation Operator

Basic principle. The oriented mutation operator is derived from the mutation operator of genetic algorithms - but can be used independently of them. The simple principle is to apply the mutation operator, but not randomly. For a given parameter, mutations are tried "both way": a very small positive number is added to the parameter and the fitness is calculated. If the fitness decreases, the orientation will be considered positive (mutation will consist of adding a positive number). If not, the number is subtracted to the parameter and the fitness calculated again. If it decreases, the orientation is considered negative. If the fitness increases or remains the same, the parameters is left as such. Once the orientation (positive or negative) is decided, the mutation operator is applied accordingly until it does not decrease the fitness anymore. By applying oriented mutation over all parameters one after another until no more decrease of the fitness can be done, the individual characterized by these parameters will rapidly move to the vicinity of the closest local minimum.

Dynamic parameter. Once the orientation is determined, the question of what value λ (mutation step, value added or subtracted from the parameter when a mutation is performed) should be used for mutation is of great importance, as it will determine both the precision and the speed of the algorithm. For optimization purposes, λ is modified dynamically as follows:

As an example, we consider an individual X characterized by n free parameters ($X = [x_1, ..., x_i, ..., x_n]$) on which the oriented positive mutation operator is applied to the parameters x_i (orientation determined using methodology described above). f(X) being the fitness function which has to be minimized, the mutation operator is applied ($x_i^{t+1} = x_i^t + \lambda$) until $f(X^{t+1}) \geq f(X^t)$, in which case we set back $x_i^{t+1} = x_i^t$. λ is then modified $\lambda = \lambda/10$ before the mutation

[1] All results presented in the document are based on calculation performed on an Intel pentium 4, 3GHz, 1000MB RAM. ANN, SOMA and back-propagation were implemented in Java and run on Ubuntu release 7.1, java 6.0_04, eclipse SDK 3.4.0.

operator is performed again. The procedure is exited when λ reach a minimal value set by the user.

This procedure allows to increase the precision when x_i approaches the vicinity of its closest minimum while keeping high speed when the algorithm starts.

2.2 Synchronized Oriented Mutations

While oriented mutation is applied on a single parameter, synchronized oriented mutation is applied on several parameters simultaneously, thus taking account the interrelated effects several parameters have when modified at the same time. First, a subset of parameters is randomly selected. Then a very small number is added or subtracted (operation randomly selected for each parameter) to each parameter of this subset and the fitness is measured. If the fitness decreased, then we consider we found a subset of parameters each being related to an orientation of mutation (positive or negative) leading to a decrease of fitness. The mutation operator is then applied accordingly simultaneously to all the selected parameters until it does not allow the fitness to decrease anymore. In the same way than described above, a dynamic mutation step λ is used.

The operations of subset selection, decision of the mutation's orientation and mutation itself are repeated until no more efficient subset that allows reduction of the fitness can be found (after a certain number of random selections and trials, here set to 100) or a maximal number of iteration (set by the user) is reached.

While genetic algorithms have been proven to be efficient in finding global minimum independently of the initial population, the method proposed here is an alternative for when the users seeks to get rapidly an acceptable solution rather than finding the optimal minimal in a more time-consuming process. The results of a run will be highly dependent of the individual minimized. But the methodology being very rapid, it allows, if diverse individuals are picked up and minimized, to find very rapidly a diverse range of acceptable solutions. Furthermore, a first rapid screening of a large number of individual using loose parameters will provide rapidly a subpopulation of promising candidates for stricter minimization.

3 Experiments

3.1 Minimizing Equation

As a first benchmark problem with multiple minimums, we use the SOMA algorithm on the following equation:

$$f(x_1,x_2,x_3) = x_1^2 + 2x_2^2 - 10sin(2x_1)sin(x_3) + 0.5cos(x_1+2x_2) + \\ x_1^2x_3^2 - 5sin(2x_1 - x_2 + 3x_3), -10 <= x_i <= 10, i = 1,2,3$$

The objective is to find x_1, x_2 and x_3 such as to minimize $f(x_1,x_2,x_3)$. The global minimum is -12.765474. Generally, searches using traditional genetics algorithms are often stuck by some local minima [2]. The strategy for applying

Table 1. Performance of SOMA on finding global minimum of equation 1 over 100 runs. The first column presents the average over all the runs between the global minimum (-12.765474) and the minimum found by SOMA after the rapid screening. The second column presents the result after applying SOMA with stricter parameters on the best individual found during screening.

minima-fitness after screening (standard deviation)	final minima-fitness (standard deviation)	calculation time in ms (standard deviation)
0.3724 (0.3727)	2.941e-5 (1.019e-4)	13.1 (11.29)

SOMA using $f(x_1, x_2, x_3)$ as a fitness function in order to find solutions is the following: In a first step a population of 30 diverse individuals $[x_1,\ x_2,\ x_3]$ is created and rapidly screened to extract its more promising member. Screening consists of applying the algorithm with loose parameters (initial λ: 1, minimal λ: 0.01, 3 iterations maximum) on each member of the population. Then SOMA is performed with much stricter parameters (initial λ: 1, minimal λ: 0.0001, 100 iterations maximum) on the individual having the lowest fitness. Results are shown on table 1. The calculation time (in milliseconds), being dependent on the operating system, is just given as an indication.

This test showed that testing a relatively small diverse population (30 individuals) allows finding very rapidly (less than 0.05s) solution in the very close vicinity of the optimal minimum (precision of 10e-4) with great confidence: solutions were found for each of the 100 runs.

3.2 Training Artificial Neural Networks

Network's structure. One of the difficulties before training an artificial neural network is to decide the structure to be used. For example, in the case of feed forward layered network, the user has to specify the number of layers as well as the number of neurons in each layer. To avoid this difficulty, we will use the following topology: A neural network will consist of an ordered array of m neurons. The first indexes of the neuron array will be occupied by input neurons while the last indexes will be occupied by output neurons. Each neuron is connected to all neurons of superior index, but with the following exceptions: input neurons are not connected to other input neurons and output neurons are not connected to other output neurons. A step of the neural network will start by feeding inputs in the input neurons. The neurons of the network then fire in order of their respecting index, the last neurons to fire being the output neurons. Using this structure the user just need to fix the number of neuron. We will refer to this structure as ordered connection feed forward network (figure 1).

Modeling trigonometric and polynomial function. As a second benchmark, the ability of training 7 neurons ordered connection feed forward network for modeling polynomials and trigonometric functions is explored. Since SOMA

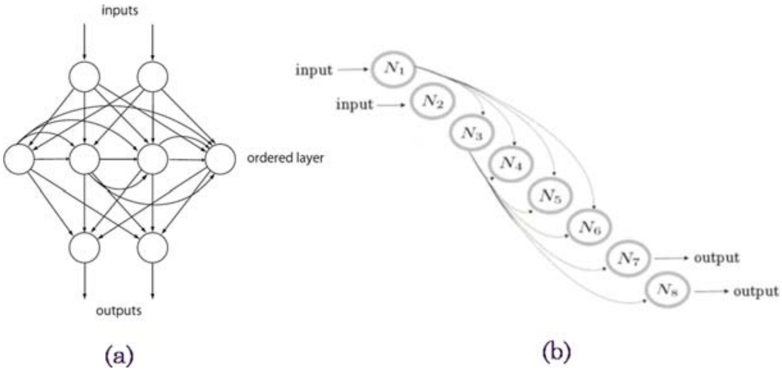


Fig. 1. Two representations of the same 8 neurons ordered connection feed forward network with 2 input neurons and 2 output neurons. (a) all the connections are represented. (b) only the connections of neurons 1 and 3 are represented. Since neuron 1 is an input neuron, it is not connected to the other input and output neurons.

allows to train simultaneously weights and parameters in the activation function, no bias neuron will be used but the sigmoid function $f(x) = 1/(1 + e^{-ax})$ will contain a parameter a subjected to learning. The results will be compared with back-propagation, using two topologies:

- The same as the one trained by SOMA (ordered connection feed forward), except that the regular sigmoid function will be used (a is fixed to 1) and a bias neuron is added.

- A one layered feed forward network with bias, with a total of 12 neurons (including input, output and bias). The number of neuron (12) was determined such as the number of free parameters (27) to be very close to the one characterizing the ordered connection structure (26 parameters).

The procedure for training is the following: polynomial functions $ax^3 + bx^2 + cx + d$ of order 1, 2 and 3 are randomly generated with $a, b, c, d : \{0, 1, 2, 3\}$. For each order, 20 functions are generated. Sine and cosine are also added to the set of functions (total of 62 functions). 120 neural networks are created and trained on each function using loose parameters (initial λ: 1, minimal λ: 0.0001, 10 iterations). From the whole population of networks, stricter parameters are applied on the 10 best individuals (initial λ: 1, minimal λ: 0.00001, 250 iterations) and the best is kept. The squared error percentage [3] between the output and f(x) is used as a fitness value, with x varying from -1 to 1 with a step of 0.125. The same procedure is applied to back-propagation, the loose parameters being a learning rate of 0.3 and 200 iterations, and the stricter parameters a learning rate of 0.2 and 800 iterations. Results are shown in table 2. We can see that contrary to SOMA, which gives correct results for all kind of functions to models, back-propagation could not train orderd connection feed forward network to model polynomial function of order 3 (increasing the number of epoch of the first and/or

Table 2. Comparative results of SOMA and BP for training ANN to model functions. Polynomial functions were grouped according to their order. SOMA was applied on ordered connection feed forward network while BP was applied on both ordered connection feed forward networks (first column) and one hidden layer feed forward network (third column).

function	BP		SOMA		BP-one layered ANN	
	average fitness	average calculation time	average fitness	average calculation time	average fitness	average calculation time
cosine	0.01	1760	0.02	25	0.01	2470
sine	0.02	1829	0.01	69	0.02	2442
order 1	0.00 (0.00)	1869.48 (75)	0.03 (0.08)	60.6 (27.71)	0.9629 (1.53)	2673.35 (396.47)
order 2	170.57 (258.90)	1948.45 (136.60)	4.07 (4.58)	241.85 (85.13)	32.12 (43.53)	3031.20 (916.22)
order 3	370.08 (385.32)	1940.30 (189.93)	15.86 (9.44)	604.45 (869.64)	91.13 (157.74)	3337.50 (610.57)

the second screening could not lead to better results, data not shown). Backpropagation applied to regular feed forward layered network led to better results, but with calculation time superior to those obtained by SOMA (the decrease of the number of epoch of the first and/or the second screening could indeed decrease this calculation time, but with drastic consequences on the quality of the model obtained, especially on polynomial functions of order 3, data not shown).

Classification: Iris Database. The ability of SOMA for training ordered connection feed forward network is further explored using the iris plant database, very well known in the pattern recognition literature [4]. The data set contains 3 classes of 50 instances each, refering to a type of iris plant. There are four numeric attributes (in cm: sepal length, sepal width, petal length, petal width) that are used as input, the output being the expected class (setosa, versicolour or virginica). There are 150 instances (50 for each class) in the database. This problem being considered easy, high speed is expected for the training phase and high confidence is expected on both learning and validation set. The methodology used for training is the following: a population of 250 diverse ordered connection feed forward networks of 9 neurons, including 4 input neurons and 3 output neurons is created. As before, the sigmoid function is used as activation function. For training, the fitness used is the squared error percentage between the output of the network and the expected output. The database is randomly separated into a learning set (70%) and a validation set (30%). During the first screening, SOMA (initial λ: 1, minimal λ: 0.001, 200 iterations maximum) is performed on each of the 250 networks. Then all the network having a classification better than 72% (on the learning set) are submitted to 20 consecutive runs of SOMA (initial λ: 1, minimal λ: 0.0001, 500 iterations maximum). The procedure is repeated 10 times using different learning and validation sets. Results are shown in table 3.

Table 3. Performance of SOMA for classifying iris type over ten run

correct classification on learning set (standard deviation)	correct classification on validation set (standard deviation)	calculation time (standard deviation)
97 [%] (1)	94 [%] (1)	29175.2 [ms] (11298.4)

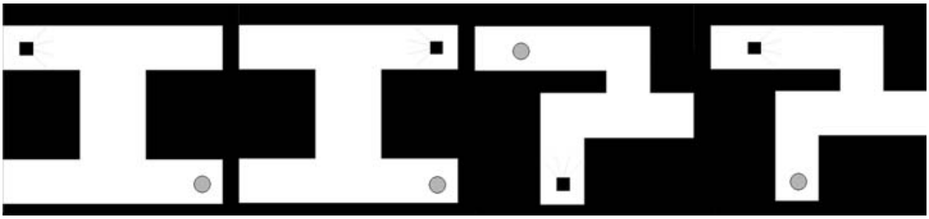


Fig. 2. The four trainging environment. The light is represented by a grey spot and the initial position of the robot by a black square.

For all ten run, very good classification percentage were found rapidly (around half a minute per run) with good generalization abilities on the validation set.

Training of robotic controller. In this section SOMA is used in a basic simulation to train an ordered connection feed forward as a robot controller. The robot, equipped with two wheels, four range sensors and two light sensors, is driven using differential kinematics. The network has eight neurons, six for the inputs (four range sensors and two light sensors) and two for the outputs (one for each wheel). The sigmoid function is used as activation function, with a free parameter a submitted to training (no bias neuron used). The network is symmetrical, insuring the left side and the right side of the robot to be treated in the same way and dividing the number of free parameters by two (40 free parameters consisting of the weights and the free parameters in the activation function). The objective of the algorithm is to train the robot to reach a light in a labyrinth using four training environments (figure 2). In a training environment, a fitness function is calculated over 6000 steps of the simulation as follow:

$$f = 2W - 3S + 2P + 10L$$

W is the sum over all the steps of the absolute difference between the right wheel speed and the left wheel speed. S refers to the sum over all the steps of the robot's speed. P is the wall penalty, consisting of the sum of the term $1/distance(wall, robot)$ over every step the robot was at a distance inferior to 20 cm of a wall. Finally the light bonus L consist of the sum of the term $10\text{x}Distance(light, robot) - 1000$ over every step the robot was at a distance inferior to 50 cm to the light. The final fitness used for training is the sum of the fitness obtained in each of the four environments. The methodology used was the following: a population of 800 networks was randomly created , screened (initial l : 1, minimal 0.001, 2 iterations maximum) and the 20 best individuals were further minimized four times (initial l : 1, minimal 0.00001, 10 iterations maximum). As a result four controllers showed to be successful in the four environments: they allowed the robot to explore the environments avoiding walls until the light was detected. The robots would then go then in its directions and remains at its vicinity.

4 Conclusion and Future Work

Despite the fact that SOMA is not population based, its speed allows to test a large number of individuals and led to good solutions for the different tasks it was tested on. Also, it showed to be more flexible than back-propagation toward the topology of the ANN used for learning. SOMA also showed to have the required characteristics for being used in the field of autonomous robotics.

Further theoritical work will have to be conducted to determine to what kind of solution space topology this approach is best adapted. Also some of its potential advantages toward genetic algorithm will be investigated. For example, it is not population-based, and can thus be applied directly to any existing controller: if such a controller is asked to perform in a slightly different environment or asked to perform a slightly different task than the one it was trained for (resulting in small modifications of the corresponding fitness function), SOMA could be rapidly applied to re-adapt the existing network.

Finally the use of SOMA as genetic operator for replacing random mutations will be considered. Similar approaches have been already been tested (using a modified mutation operator, methodology often referred as Baldwin effect) and proved successful [2]. SOMA could be used in the same way with the advantage of modifying different parameters simultaneously.

References

1. Yao, X.: Evolving artificial neural networks. Proc. IEEE 87(9), 1423–1447 (1999)
2. Sun, Y., Deng, F.Q.: Baldwin effect self adaptive generalized genetic algorithm. In: 8th International conference on control automation, robotics and vision Kumming, ICARCV, China, pp. 242–247 (2004)
3. Prechelt, L.: PROBEN1 - A set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, Universitaet Karlsruhe (1994)
4. 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`

Bioinspired Parameter Tuning of MLP Networks for Gene Expression Analysis: Quality of Fitness Estimates vs. Number of Solutions Analysed

André L.D. Rossi[1], Carlos Soares[2], and André C. P. L. F. Carvalho[1]

[1] Computer Science Department - University of São Paulo, Brazil
{alrossi,andre}@icmc.usp.br
[2] LIAAD-INESC Porto LA - University of Porto, Portugal
csoares@fep.up.pt

Abstract. The values selected for the free parameters of Artificial Neural Networks usually have a high impact on their performance. As a result, several works investigate the use of optimization techniques, mainly metaheuristics, for the selection of values related to the network architecture, like number of hidden neurons, number of hidden layers, activation function, and to the learning algorithm, like learning rate, momentum coefficient, etc. A large number of these works use Genetic Algorithms for parameter optimization. Lately, other bioinspired optimization techniques, like Ant Colony optimization, Particle Swarm Optimization, among others, have been successfully used. Although bioinspired optimization techniques have been successfully adopted to tune neural networks parameter values, little is known about the relation between the quality of the estimates of the fitness of a solution used during the search process and the quality of the solution obtained by the optimization method. In this paper, we describe an empirical study on this issue. To focus our analysis, we restricted the datasets to the domain of gene expression analysis. Our results indicate that, although the computational power saved by using simpler estimation methods can be used to increase the number of solutions tested in the search process, the use of accurate estimates to guide that search is the most important factor to obtain good solutions.

1 Introduction

Artificial Neural Networks (ANNs) [10] have provided powerful tools for several Pattern Recognition tasks. It is common sense, however, that the performance achieved by ANNs is largely affected by the appropriate choice of values of their free parameters. These parameters are usually tuned by trial and error, which is a subjective and time consuming process. Several works try to overcome this problem by using optimization techniques, mainly metaheuristics, for the selection of values related to the network architecture (e.g., number of hidden neurons, number of hidden layers, activation function) and to the learning algorithm (e.g., learning rate, momentum coefficient). A large number of these parameter optimization works use Genetic Algorithms (GA) (e.g., [14,2,13,3,19]). Lately, other bioinspired optimization techniques, like Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), among others, have been successfully used. In [5], PSO was used to optimize the parameters of different

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 252–259, 2009.

connectionist paradigms. A modified PSO was applied by [7] to simultaneously tune structure and connection weights of ANNs. A learning algorithm based on an artificial immune system was used by [12] to design fuzzy-neural networks. Although bioinspired optimization techniques have been successfully adopted to tune neural networks parameter values, there is a lack of works comparing the use of these bioinspired techniques in a controlled scenario.

In previous work, we have compared the performance of different bioinspired optimization techniques on the choice of the parameters for Multi-Layer Perceptron (MLP) networks [10] and their learning algorithm parameters [16]. Here, as in the previous paper, MLP networks have their number of hidden neurons and training parameters adjusted by the bioinspired optimization techniques. The networks obtained are trained by the backpropagation algorithm with momentum [17]. Four different bioinspired optimization techniques were applied on the problem of ANN parameter tuning. These techniques are: Ant Colony Optimization (ACO) [6], Particle Swarm Optimization (PSO) [18], Clonal Selection Algorithm (CSA) [4] and GAs [8].

In order to reduce variability caused by using datasets from different domains in the experimental results, we restricted the datasets to one particular domain, gene expression analysis. The choice of this domain was motivated by a particular characteristic found in datasets from this domain: small number of examples (tens) with a very large number of features (thousands).

In order to maximize the quality of the estimates of fitness of each solution, an internal 10 times/10-fold cross-validation procedure was used to assess the error of the MLP. This approach is obviously very computationally expensive. In this paper, we investigate the effect of reducing the number of experiments done for estimating fitness on the trade-off between the loss in the quality of those estimates and the gain obtained by searching more solution in the time which is saved.

This paper is organized as follows. Section 2 presents the methodology followed in the experiments. Section 3 shows the experimental results and their analysis. The main conclusions of this paper are discussed in Section 4.

2 Experimental Methodology

In this section, we describe the methodology used to compare the bioinspired optimization techniques on the problem of tuning the parameters of the MLP.

2.1 Optimization Problem

As stated earlier, the goal of this work is to tune the parameters of MLP networks. Three parameters were considered: the number of neurons in the hidden layer (γ), the learning rate (η) and the momentum coefficient (μ). A single hidden layer was used. This enables the representation of a range of functions that is sufficiently large for the purpose of this study, which is not to have the best possible architecture, but to compare different parameter tuning approaches. The *AMORE*[1] R package implements the backpropagation algorithm with momentum. The domains of the parameters were limited as follows: $\gamma \in \{2, 10^2\}$, $\eta \in \{5^{-2}, 1\}$ and $\mu \in \{0, 1\}$.

[1] http://cran.r-project.org/web/packages/AMORE/index.html

The fitness of a solution on a given dataset is computed as follows. A MLP network with the number of hidden units set to the value of γ in the solution, is assigned initial weights randomly. Then, it is trained using the backpropagation algorithm with parameters η and μ set to the corresponding values in the solution.

The final weights obtained with the backpropagation algorithm with a fixed set of parameters may vary with the initial weights. This implies that the performance of a given individual will probably also vary with the set of initial weights. In our earlier work, to reduce the effect of this variance, we used N-fold cross-validation with r repetitions of the learning process for each fold [16]. The dataset is divided into N_f subsets. We then use each of these subsets in turn as a fitness estimation set, and the remaining data as training set. For each fold, a MLP was learned using r different initial weights subsets and the mean error is assigned to the fold. The fitness of the solution is computed as the mean of the error rates computed in the N_f folds.

In this paper, we analyse the effect of replacing the average of the $r \times N_f$ results with the results of a single run of the algorithm as the estimate of the fitness, both in terms of the reduction of the computational cost of estimation and on the quality of the optimization process. We will refer to the first fitness estimation process as 10/10, because we have used $r = 10$ and $N_f = 10$, and to the second as 1/1.

2.2 Optimization Methodology

To ensure, as much as possible, that the techniques are tested under the same conditions, we analysed the number of evaluations that each algorithm performs to find its best solution. To evaluate the solutions obtained by the optimization methods, the error rate of the best solution found by each one is estimated using a separate test set. The reliability of the test error estimates is affected by three factors: the random nature of the optimization techniques, the random sample of the data into training and test sets and the random initial weights of MLP network. We address this three issues by estimating the test error as the average of 30 runs, by adopting a 10-fold cross-validation procedure for estimation of the test error and by training each network ten times with random initial weights for each solution.

This experimental setup requires a large number of executions of the backpropagation algorithm. In the case of the 10/10 process, the execution of a single test fold optimization process depends on the number of folds used for fitness estimation (N_f) and the number of repetitions r for each fold. Finally, it depends on the number of individuals tested by each bioinspired technique, s, yielding a total of $N_f \times r \times s$ executions. For $N_f = 10$, $r = 10$ and $s = 3\,000$, the backpropagation algorithm would be executed 300 000 times. To reduce the number of executions of the backpropagation algorithm, the number of different values considered is reduced by discretizing the values of the continuous parameters, η and μ. This is achieved by rounding the corresponding values to the nearest adequate fractions. The fitness of every combination of parameter values is stored in an appropriate data structure. When evaluating an individual, the fitness value previously stored is retrieved without the need to execute the backpropagation algorithm once more.

In the experiments reported in this paper, the values of the η and μ were rounded to fractions of 0.05 and 0.1, respectively. Therefore, the domain of values for the

η parameter is $\{0.05, 0.1, \ldots, 1\}$ and for the μ parameter, it is $\{0, 0.1, \ldots, 1\}$. Additionally, the numbers of hidden neurons, γ, was rounded to even values, namely $\{2, 4, \ldots, 100\}$. This means that the search space of the optimization techniques contains a total of $20_{(\gamma)} \times 11_{(\eta)} \times 50_{(\mu)} = 11\,000$ different combinations of parameter values.

Finally, it is important to assess whether the optimization techniques are really doing useful search or not. For this purpose, their results are compared with those obtained with two simple baseline strategies. The first one is a single MLP obtained using the default parameter values of WEKA, i.e., $\gamma = (A + C)/2$, $\eta = 0.3$ and $\mu = 0.2$, where A and C are the number of attributes and classes of the datasets, respectively. In the results reported below, we refer to this method as *default*. The second baseline consists of randomly generating s combinations, where s is the approximate number of individuals tested by each technique. Given the nature of this method, 30 runs were carried out and its performance is computed as the mean of the 30 results obtained. This strategy is referred to as *random*.

3 Experimental Results

Four binary-class gene expression datasets were used in the experiments: colon [1], glioma [15], pancreas [11] and leukemia [9]. Class distribution on these datasets is $(Class1/Class2)$: 40/22, 28/22, 24/25 and 49/51, respectively. The attributes are all numerical, each one of them representing a single gene.

Tables 1 to 4 present the mean error rate of the best ANNs found by each optimization technique, on the fitness set and on the test set using both the 10/10 and the 1/1 fitness estimation procedures. The standard deviation (in percentage) for 30 runs of the techniques is presented in parenthesis. The number of different individuals (combinations) that is tested by each bioinspired technique is also presented.

We start by comparing different parameter tuning techniques obtained with the 10/10 fitness estimation process (Section 3.1). The parameter values used for the bioinspired techniques are described in [16]. Then, we present the results of replacing the estimates of fitness using a 10 times/10 fold cross-validation procedure (10/10) with a simpler process, which uses only one of those results (1/1) (Section 3.2).

Table 1. Results on the Colon dataset

	Fitness			**Test**			**Dif. Solutions**		
Tech.	**10/10**	**1/1**	**Dif.**	**10/10**	**1/1**	**Dif.**	**10/10**	**1/1**	**Dif.**
ACO	12.70 (0.1)	0.06 (0.3)	12.64	18.09 (1.0)	19.66 (2.2)	-1.58	1546	8798	-7252
CSA	13.90 (0.2)	0.00 (0.0)	13.90	15.84 (1.8)	18.33 (3.1)	-2.49	1425	893	532
GA	13.66 (0.2)	1.11 (1.2)	12.55	16.59 (1.6)	17.30 (2.7)	-0.71	253	4426	-4174
PSO	13.45 (0.1)	1.06 (1.0)	12.39	16.50 (1.7)	17.53 (2.4)	-1.03	1517	1289	228
Default	19.81 (-)	18.33 (-)	1.48	17.86 (-)	17.86 (-)	◇	◇	◇	◇
Random	13.23 (0.2)	0.00 (0.0)	13.23	16.68 (1.6)	17.49 (1.7)	-0.81	2623	9987	-7365

Table 2. Results on the Glioma dataset

	Fitness			Test			Dif. Solutions		
Tech.	**10/10**	**1/1**	**Dif.**	**10/10**	**1/1**	**Dif.**	**10/10**	**1/1**	**Dif.**
ACO	12.31 (0.2)	2.00 (0.0)	10.31	16.04 (1.7)	22.14 (3.3)	-6.10	1552	8800	-7248
CSA	12.80 (0.3)	2.00 (0.0)	10.80	17.50 (2.5)	19.25 (3.4)	-1.75	1481	1452	29
GA	13.04 (0.1)	3.20 (1.0)	9.84	14.54 (0.9)	19.30 (2.6)	-4.75	317	4488	-4171
PSO	12.79 (0.1)	2.93 (1.0)	9.86	14.02 (1.3)	19.69 (3.4)	-5.66	1477	1532	-55
Default	18.02 (-)	16.00 (-)	2.02	15.80 (-)	15.80 (-)	◇	◇	◇	◇
Random	12.60 (0.2)	2.00 (0.0)	10.60	17.47 (2.1)	19.42 (3.6)	-1.95	2622	9986	-7364

Table 3. Results on the Leukimia dataset

	Fitness			Test			Dif. Solutions		
Tech.	**10/10**	**1/1**	**Dif.**	**10/10**	**1/1**	**Dif.**	**10/10**	**1/1**	**Dif.**
ACO	12.83 (0.1)	1.00 (0.0)	11.83	20.34 (0.9)	21.16 (1.8)	-0.82	1608	8792	-7184
CSA	13.92 (0.2)	1.00 (0.0)	12.92	19.45 (1.3)	22.08 (2.3)	-2.63	1384	739	646
GA	13.88 (0.3)	1.00 (0.0)	12.88	20.33 (0.9)	22.82 (2.9)	-2.48	256	4584	-4328
PSO	13.33 (0.2)	1.00 (0.0)	12.33	19.72 (1.0)	22.82 (1.9)	-3.10	1557	894	663
Default	18.94 (-)	9.00 (-)	9.94	23.03 (-)	23.03 (-)	◇	◇	◇	◇
Random	13.21 (0.1)	1.00 (0.0)	12.21	19.22 (1.2)	22.64 (2.1)	-3.43	2623	9988	-7365

Table 4. Results on the Pancreas dataset

	Fitness			Test			Dif. Solutions		
Tech.	**10/10**	**1/1**	**Dif.**	**10/10**	**1/1**	**Dif.**	**10/10**	**1/1**	**Dif.**
ACO	12.42 (0.2)	0.67 (0.8)	11.75	21.60 (1.5)	19.47 (2.1)	2.14	1615	8796	-7181
CSA	13.96 (0.2)	0.50 (0.8)	13.46	20.30 (1.8)	20.92 (2.3)	-0.62	1536	1159	377
GA	14.34 (0.5)	1.44 (0.6)	12.89	18.88 (1.8)	20.54 (2.0)	-1.66	416	5408	-4992
PSO	13.39 (0.3)	1.22 (0.7)	12.17	19.04 (1.6)	21.27 (2.2)	-2.23	1775	1471	304
Default	17.76 (-)	19.67 (-)	-1.91	17.33 (-)	17.33 (-)	◇	◇	◇	◇
Random	13.25 (0.2)	0.44 (0.7)	12.80	20.00 (1.7)	21.64 (2.4)	-1.63	2623	9989	-7366

3.1 Comparison of Optimization Methods Using the 10/10 Fitness Estimation Process

It is possible to observe that the bioinspired techniques generally obtain better results than the default method. This means that the search made by these techniques is, in fact, working as expected. On the other hand, they are similar to the results obtained with the random search method. This indicates that parameter tuning problems that these datasets pose are not hard. Statistical tests confirm these observations [16]. The comparison between the different optimization techniques shows that ACO achieves the best results. It obtains the lowest fitness error on all datasets. Nevertheless, it is not possible to clearly identify the best algorithm in terms of the test error.

The number of different combinations tested by each technique, presented in Tables 1-4, provides information concerning the compromise between exploration and exploitation. As stated earlier, the maximum number of individuals that each technique is able to test is determined by its parameters. In our experiments, the ACO, PSO and GA could test approximately 3 000 individuals, while the CSA, due to the cloning process, could test approximately 7 000 individuals. The size of the search space is 11 000 different combinations. The proportion of different combinations tested by ACO, PSO and CSA indicate a good balance between exploration and exploitation of the search space. On the other hand, the GA explored a much smaller proportion of different combinations. This occurs even though elitism, which increases exploitation, was not used. This may be due to the operators that were used and also to the low mutation probability. Nevertheless, its results are comparable to the results obtained with the other techniques. This means that there is not a clear correlation between the amount of exploration and the quality of the solutions obtained.

3.2 Effect of Simplifying the Fitness Estimation Process

Given that the computational effort of the 1/1 method is approximately 100 times less than that of 10/10, we have allowed the optimization methods to evaluate more solutions in the former case (approximately 10 times more solutions).

Tables 1-4 show that the test error obtained with 10/10 is systematically lower than the error obtained with 1/1. On the other hand, the fitness error obtained with 1/1 is significantly lower. In fact, it is almost zero, which indicates that overfitting may be occurring. Additionally, it also indicates that even if more than 10 times the solutions were tested using 1/1 than with 10/10, the solution would probably not improve.

According to these results, there is a trade-off between the quality of the estimates of fitness and the number of solutions tried out. However, it is more important for the search process to obtain accurate estimates. However, we note that 10/10 and 1/1 represent extreme scenarios and a better trade-off could be obtained with intermediate ones, such as, 1/10, i.e., 10 repetitions of a training/test procedure.

4 Conclusions

Recently, a comparison between four different bioinspired optimization techniques on the problem of tuning the parameters of MLP networks for classification tasks has been published [16]. The techniques considered were Ant Colony Optimization, Particle Swarm Optimization, Clonal Selection Algorithm and Genetic Algorithms. Although these techniques are commonly used on this task, no comparative study had been carried out before that. The focus was on a single domain, gene expression analysis, to reduce the variation which would be introduced by testing on datasets from different domains and also because Bioinformatics is an important area of application of learning methods.

This paper extends that previous work by investigating the relation between the quality of the estimates of the fitness of a solution used during the search process and the quality of the solution obtained by the optimization method. An empirical study is performed to assess the effect of reducing the number of experiments done for estimating fitness on the trade-off between the loss in the quality of those estimates and the gain obtained by searching more solutions in the time which is saved.

Our results clearly show that such a compromise exists. However, they also show that the quality of the fitness estimates is more important for the final solution than making a longer search. However, we only tested two extreme settings so, further work is required to better understand the trade-off.

Acknowledgments. The authors acknowledge the support from FAPESP, CNPq and FCT (project Triana - POCT/TRA/61001/2004 - and Programa de Financiamento Pluri-anual de Unidades de I&D).

References

1. Alon, U., Barkai, N., Notterman, D.A., Gish, K., Ybarra, S., Mack, D., Levine, A.J.: Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc. of the National Academy of Sciences 96(12), 6745–6750 (1999)
2. Blanco, A., Delgado, M., Pegalajar, M.C.: A real-coded genetic algorithm for training recurrent neural networks. Neural Networks 14(1), 93–105 (2001)
3. Castillo, P.A., Merelo, J.J., Arenas, M.G., Romero, G.: Comparing evolutionary hybrid systems for design and optimization of multilayer perceptron structure along training parameters. Information Sciences 177(14), 2884–2905 (2007)
4. Castro, L.N., Von-Zuben, F.: Learning and optimization using the clonal selection principle. IEEE Transactions on Evolutionary Computation 6(3), 239–251 (2002)
5. Chen, Y., Abraham, A.: Hybrid learning methods for stock index modeling. In: Kamruzzaman, J., Begg, R.K., Sarker, R.A. (eds.) Artificial Neural Networks in Finance, Health and Manufacturing: Potential and Challenges, Idea Group Inc. Publishers, USA (2006)
6. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization: Artificial ants as a computational intelligence technique. IEEE Comp. Intel. Mag. 1(4), 28–39 (2006)
7. Gao, L., Zhou, C., Gao, H.-B., Shi, Y.-R.: Credit scoring model based on neural network with particle swarm optimization. In: Proc. of the Sec. Int. Conf. on Advances in Natural Computation, pp. 76–79. Springer, Heidelberg (2006)
8. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
9. Haslinger, C., Schweifer, N., Stilgenbauer, S., Dohner, H., Lichter, P., Kraut, N., Stratowa, C., Abseher, R.: Microarray Gene Expression Profiling of B-Cell Chronic Lymphocytic Leukemia Subgroups Defined by Genomic Aberrations and VH Mutation Status. J. of Clinical Oncology 22(19), 3937–3949 (2004)
10. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice-Hall, Englewood Cliffs (1999)
11. Ishikawa, M., Yoshida, K., Yamashita, Y., Ota, J., Takada, S., Kisanuki, H., Koinuma, K., Choi, Y.L., Kaneda, R., Iwao, T., Tamada, K., Sugano, K., Mano, H.: Experimental trial for diagnosis of pancreatic ductal carcinoma based on gene expression profiles of pancreatic ductal cells. Cancer Science 96(7), 387–393 (2005)
12. Kim, D.H., Abraham, A.: Optimal learning of fuzzy neural network using artificial immune algorithm. Neural Network World 18(2), 147–170 (2008)
13. Lacerda, E.G.M., Carvalho, A.C.P.L.F., Ludermir, T.B.: Model selection via genetic algorithms for rbf networks. J. of Intelligent and Fuzzy Systems 13(2-4), 111–122 (2002)

14. Leung, F.H.F., Lam, H.K., Ling, S.H., Tam, P.K.S.: Tuning of the structure and parameters of a neural network using an improved genetic algorithm. IEEE Transactions on Neural Networks 14(1), 79–88 (2003)
15. Nutt, C.L., Mani, D.R., Betensky, R.A., Tamayo, P., Gregory Cairncross, J., Ladd, C., Pohl, U., Hartmann, C., McLaughlin, M.E., Batchelor, T.T., Black, P.M., von Deimling, A., Pomeroy, S.L., Golub, T.R., Louis, D.N.: Gene expression-based classification of malignant gliomas correlates better with survival than histological classification. Cancer Research 63(7), 1602–1607 (2003)
16. Rossi, A.L.D., Carvalho, A.C.P.L.F., Soares, C.: Bio-inspired parameter tunning of mlp networks for gene expression analysis. In: International Conference on Hybrid Intelligent Systems, pp. 435–440 (2008)
17. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In: Parallel distributed processing: explorations in the microstructure of cognition, foundations, vol. 1, pp. 318–362. MIT Press, Cambridge (1986)
18. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: Proc. of the IEEE Int. Conference on Evolutionary Computation, pp. 69–73. Anchorage, Alaska (1998)
19. Tsai, J.-T., Chou, J.-H., Liu, T.-K.: Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm. IEEE Transactions on Neural Networks 17(1), 69–80 (2006)

Sample Filtering Relief Algorithm: Robust Algorithm for Feature Selection

Thammakorn Saethang[1], Santitham Prom-on[2], Asawin Meechai[3],
and Jonathan Hoyin Chan[4,*]

[1] School of Information Technology and School of Bioresources and Technology
[2] Pilot Plant Development and Training Institute
[3] Department of Chemical Engineering and Biological Engineering Program
[4] School of Information Technology
King Mongkut's University of Technology Thonburi, Bangkok, Thailand
Tel.: (662) 470-9819; Fax: (662) 872-7145
jonathan@sit.kmutt.ac.th

Abstract. Feature selection (FS) plays a crucial role in machine learning to build a robust model for either learning or classification from a large amount of data. Among feature selection techniques, the Relief algorithm is one of the most common due to its simplicity and effectiveness. The performance of the Relief algorithm, however, could be dramatically affected by the consistency of the data patterns. For instance, Relief-F could become less accurate in the presence of noise. The accuracy would decrease further if an outlier sample was included in the dataset. Therefore, it is very important to select the samples to be included in the dataset carefully. This paper presents an effort to improve the effectiveness of Relief algorithm by filtering samples before selecting features. This method is termed Sample Filtering Relief Algorithm (SFRA). The main idea of this method is to discriminate outlier samples out of the main pattern using self organizing map (SOM) and then proceed with feature selection using the Relief algorithm. We have tested SFRA with a gene expression dataset of *interferon-*α(IFN-α) response of Hepatitis B patients that contains outlier data. SFRA could successfully remove outlier samples that have been verified by visual inspection by experts. Also, it has better accuracy in separating the relevant and irrelevant features than other feature selection methods considered.

1 Introduction

Feature selection plays a crucial role in machine learning because irrelevant or noisy features could affect the discriminative power of the algorithm or model. Selecting only a minimal set of informative and relevant features could improve the robustness of algorithms or models for learning parameters, classifying samples, or predicting responses from a large amount of data. The fundamental motivation behind almost every feature selection techniques is the *curse of dimensionality*, which is the condition that the system or algorithm has to process large number of features with small

[*] Corresponding author.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 260–267, 2009.

number of samples. Feature selection can alleviate this problem by removing irrelevant and redundant features. The smaller number of consistent features would result in faster and more reliable data analysis [1].

The machine learning algorithms and models such as decision tree, artificial neural network, Bayesian network and other classification systems are error-prone in the presence of irrelevant and noisy features [2]. Feature selection can enhance the robustness of these algorithms by searching for the feature subset which is most pertinent and best representing the data. For instance, in [3], the combination between feature selection techniques with ensemble neural network showed better accuracy in cancer gene expression classification. Three feature selection methods (rank sum test, PCA, *t*-test) were evaluated to determine the statistically significant features and those features were used to construct an artificial neural network for classification of sample class. Another example is the evaluation of similarity measure including Pearson's correlation coefficient, Spearman's correlation coefficient, Euclidean distance, and cosine coefficient to find essential features before using the ensemble neural network for classification [4]. The results from both studies showed that using feature selection yielded higher prediction accuracy than without.

Comparative studies of feature selection were conducted in [5, 6] to assess the classifiers' performance. However, these works did not find a single best classifier for all datasets used. This is because each dataset has some specific properties that may affect classifier performance differently. Nonetheless, their findings indicated that the overall accuracy was generally good for datasets with a small number of classes. However, the prediction accuracy was dramatically lower for the datasets with a large number of classes. Feature selection in classification has been used in various bioinformatics related applications such as cancer classification [7], diagnosis and prognosis of cancer [8], and in drug discovery [9].

In general statistical analyses, the performance of feature selection methods could be deteriorated from the presence of outliers in the dataset. The outlier can be either a feature outlier or a sample outlier which is defined by specific properties of each data. Outliers themselves are often the special points of interest in many practical situations and their identification is the main purpose of this investigation. In bioinformatics study, the classical statistical methods based on mean and covariance matrix may not be able to detect multivariate outliers [10]. Therefore, identifying such multivariate outliers is important for improving consistency of the pattern of data which would result in better classification power.

Most outlier identification methods generally focus on removing feature outliers [10-12]. However, sample outlier profile identification or mostly called sample filtering has not been studied. For example, a method called Cancer Outlier Profile Analysis (COPA) was proposed for detection of the profile outlier sample of microarray gene expression data based on principle of chromosomal rearrangement [13]. COPA was used to calculate cancer association score for each gene at different percentiles. Genes with high score are considered as important genes that may be related to cancer. In addition, a measurement called outlier sums was developed. It is based on interquatile range (IQR), and focuses on the group of samples that defined as a case group only (disease group). A sample would be identified as an outlier if their score is higher than the threshold. These methods defined the presence of outliers as an important factor for identification of significant genes. On the other hand, in terms of classification, an outlier is the unwanted information that could affect the consistency of the model and should be removed from the data. This paper, therefore, uses the later definition of outlier and focuses on improving feature selection techniques by removing the outlier before training or analyzing the data [14].

Among existing feature selection techniques, the Relief algorithm is one of the most successful techniques because of its simplicity and effectiveness [15]. The key goals of the Relief algorithm are to estimate the relevancy of features according to how well their values are distinguishable among samples that are near to each other, and to estimate features having intense dependencies between samples for selection of feature subsets. Nevertheless, the original Relief is not robust against incomplete or outlier data and is limited to only two-class problems [16].

There are few extensions of Relief algorithm that have been developed to overcome these problems. For example, both Relief-F and Relief-F regression can be used to solve the multi-class problems by adjusting the function that calculate the weigh for each feature [17]. Iterative Relief algorithm (I-Relief) is a modification based on the Expectation-Maximization (EM) algorithm to estimate the optimal solution and improve the solution convergence [15]. Although, in terms of convergence, I-Relief is better than Relief-F, the former also requires more computational time. Therefore I-Relief is not practical when the data contains a large number of features such as microarray data. Therefore, this study uses Relief-F as a basis for feature selection techniques for the microarray dataset studied.

Since Relief-F algorithm is not robust in the presence of outliers, outlier filtering techniques should be included in the algorithm. This paper proposes a modified Relief algorithm, termed Sample Filtering Relief Algorithm (SFRA), to remove outliers before selecting features. First, several outlier filtering methods were tested. Then, the most accurate method was incorporated into the Relief-F algorithm. Afterward, SFRA was used to find the significant features. These features were then used in the classification task for biomarker identification. The pseudo-code for the overall methodology of SFRA is given below.

Pseudo code of Sample Filtering Relief Algorithm (SFRA)

```
Detect and remove outlier sample(s) by SOM
Set all weights W[A] = 0.0
for i = 1 to $m$ do begin
 randomly select sample $R_i$
 find $k$ nearest hit $H_j$
   for each class $C \neq class(R_i)$ do
     find $k$ nearest miss $M_j(C)$ from class C
     for A = 1 to all feature do
       $W(A) = W(A) - \sum_{j=1}^{k} diff(A, R_i, H) / (m.k) + \sum_{C\neq class(R_i)}^{k} [ \frac{P(C)}{1-P(class(R_i))} \sum_{j=1}^{k} diff(A, R_i, M_j(C)) ] / (m.k)$
     end
   end
end
```

2 Hepatitis B Dataset

The dataset used in this study is a microarray dataset of the differential gene expression study of hepatitis B patients that responded to drug treatment and those who did not respond to drug treatment. Specifically, the dataset consists of *interferon-α(*IFN-*α)* response of Hepatitis B patients. The dataset is a gene expression microarray data of peripheral blood mononuclear cell (PBMC) of patients using Illumina HumanRef-8 Expression BeadChips microarray. There are 22,184 genes with 23 cell line samples. The samples can be separated into 2 classes, 12 samples in SVR (sustained virological response) and 11 samples in NR (non-response). The SVR and NR classes are the groups of patients that responded and not responded to the treatment of IFN-α respectively. One sample (no. 25) was excluded from the dataset because of mislabeling. The goal of analysis for this dataset is to identify biomarkers which play a role in IFN-α response of hepatitis B patients from the different gene expression profile pattern between groups.

The dataset was first normalized by using quantile normalization. The significant genes were then identified by using *t*-test (p<0.05 and fold change>1.3). A total of 183 significant genes that differentially expressed were found. *Z*-scores of these significant genes were then computed to standardize the expression profile across all genes. Finally, the gene expression profiles of both classes were calculated by K-means clustering (K=5) to determine expression profile pattern (see Fig.1).

From Fig.1, sample 34 is visually distinguishable in SVR class from others in the group. For NR, samples 12, 37, and 39 seem to have contradictory profiles. Therefore, by using human "expert" visualization, samples 34, 12, 37 and 39 were considered to be outlier samples. These outliers could interfere with further analysis, especially in the selection of significant genes or identification of the biomarkers.

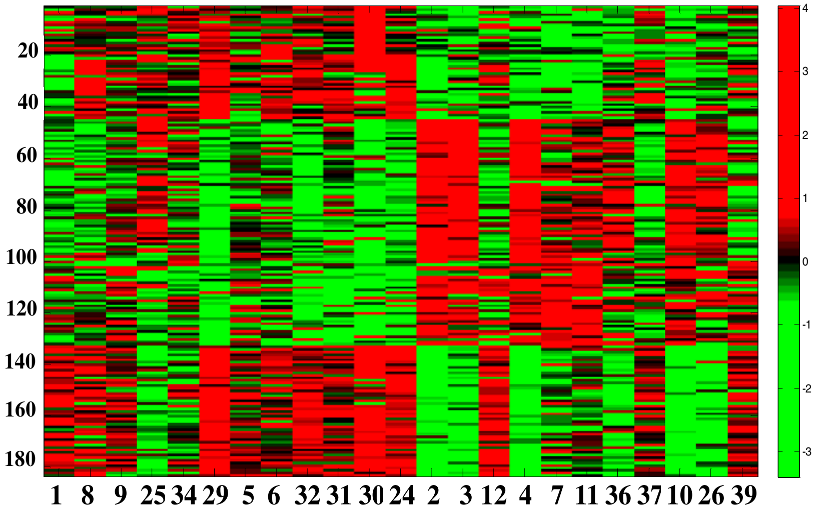


Fig. 1. Heatmap of Hepatitis B patient gene expression profiles. (The 12 samples to the left of "2" are in SVR class, and those to the right are in NR class).

3 Methods for Outlier Filtering

To find a suitable technique to remove outlier profiles, we evaluated a number of methods including *t*-test, inter-quartile range (IQR) score, principal component analysis (PCA), hierarchical clustering (HCL), and self-organizing map (SOM). By using *t*-test, the *p*-value for each sample was calculated from all gene expression value in that sample. If *p*-value was lower than the significant level of α=0.05, the sample would be identified as an outlier sample. The IQR score was calculated from the expression range from first to third quartile of the sample. The score was calculated from the number of samples that has values outside 1.5 times the IQR of other samples. Any sample with a higher score than threshold (at 95% confidence level) was considered to be an outlier sample. For PCA, we used median centering mode and the number of neighbors for k-nearest neighbors (KNN) imputation was set to 10. The goal of PCA is to reduce large dimensionality of dataset to lower dimensions for analysis, which are adequate in representing the data. For HCL, it is the separating of a dataset into subsets called clusters; the data in each subset share some common trait according to some defined distance measure. We applied average linkage clustering and used Euclidean distance matrix in our work.

SOM is a popular artificial neural network based on unsupervised learning. It consists of neurons organized on a regular low-dimensional grid called a map, usually in rectangular or hexagonal configuration. The training parameters such as the training length, the training rate, and the size of the updating neighborhood are user defined parameters. After the network is stable, these reference vectors are used to group the data points into clusters based on the closeness of the data points to the reference vectors [18].

4 Experimental Results

4.1 Outlier Filtering

Experiments were conducted to compare the performance of all mentioned methods. The results from *t*-test could not detect any outlier samples because the resulting *p*-value for all samples was equal to 0, at α = 0.05, indicating all samples were outlier samples. The IQR score indicated sample 30 as an outlier sample with the highest score. The same result was also found with HCL and PCA methods. Results of HCL showed that sample 30 was separated from other samples when clustering using all samples and only SVR class samples. However, clustering result of only NR class did not indicate any outlier sample. Moreover, when performing PCA using all samples and only SVR class sample, the results showed that sample 30 was strongly an outlier sample. Also, no outlier was detected in the NR class samples.

In contrast, SOM could detect most of the outlier samples in NR class and all in SVR class (see Table 1). However, SOM also detected sample 30 as an outlier. Even though it was not considered to be an outlier from the pre-defined visualization of gene expression profile, it has a contrasting expression profile to others. This sample has very high expression level which is obvious from the very high intensity of red color. Therefore, sample 30 may indeed be an outlier sample in this context.

From the results, SOM was better in indicating outlier samples (see Table 2), but appropriate parameters needed to be defined. For this study, the grid dimension was set to 3×3 ($\sqrt{n}\times\sqrt{n}$, where n is the number of samples), the learning rate (α) was set to 0.005, and 2000 iterations were used. Furthermore, the use of rectangular topological, Euclidean distance function, and Gaussian neighborhood function was required to generate the result clusters that could separate an outlier sample from other samples.

Table 1. Summary of SOM results by class

Cluster	SVR class		NR class	
	Sample(s) in the cluster	Average Euclidean distance	Sample(s) in the cluster	Average Euclidean distance
1	1, 6, 8	15.68507	4	16.20966
2	-	-	7	15.84978
3	5, 9, 31	15.14084	11	15.42758
4	-	-	36	15.64289
5	-	-	-	-
6	34	16.13726	3	14.79307
7	24, 29	14.70772	26	16.27777
8	32	14.27062	12*, 37*, 39*	17.43354*
9	30*	18.52925*	2, 10	15.70184

* indicate an outlier sample cluster at 95% confident.

Table 2. Comparison of methods used to detect outlier samples in gene expression profile

Method	Selected outlier sample	Matched outlier sample	Matched outlier (from 4)	Not Matched outlier	Computational time used (10^{-3} second)
t-test	all samples	12,34,37,39	4	18	~980
IQR score	30	-	-	1	~300
HCL	30	-	-	1	~484
PCA	30	-	-	1	~492
SOM	12, 37, 39, 30	12, 37, 39	3	1	~290

One of the pre-defined outlier samples, sample 34, could not be detected by any method that we used. This means the gene expression profile of this sample may not be significantly different from others in the same class. Therefore visualization alone, without any indicator values, may lead to incorrect outlier identification.

4.2 Classification Results

After detection and removal of outlier samples (12, 30, 37, 39), the data was normalized by *Z*-score and then Relief-F was applied to find the significant genes, followed by the classification task. The accuracy of SFRA was compared with the original Relief-F without outlier sample filtering. The most appropriate parameter *k* for Relief-F used in this work was set to *roundup*($\log_2 n$) where *n* is the number of samples [19]. The result of classification task is shown in Table 3. In most cases, SFRA showed higher accuracy than original Relief-F when using K-nearest neighbors (KNN) or Naïve Bayes (NB) as a classifier. The accuracy was 100% when the

Table 3. Comparison of prediction accuracy (using LOOCV) between original Relief and SFRA algorithms. (Using KNN, with number of neighbors equal to 3, and NB as classifiers).

Significant gene(s)	Original Relief-F (outlier not filtered)		SFRA (outlier filtered by SOM)	
	KNN	NB	KNN	NB
1	77.27	81.82	100	100
10	95.45	95.45	94.44	88.89
20	90.91	81.82	100	94.44
50	90.91	81.82	100	94.44
100	90.91	81.82	94.44	88.89
200	86.36	81.82	88.89	88.89
2000	81.82	77.27	88.89	83.33

number of significant gene(s) was 1, 20, and 50 for KNN and only 1 for NB. Therefore, our results indicate that SFRA performs better than the original Relief-F. After the identification of the highly informative gene(s), further analysis will be required clinically to find potential biomarker(s).

5 Conclusions and Future Research

This study proposes a modified Relief algorithm, termed Sample Filtering Relief Algorithm (SFRA), which embeds SOM as an outlier filtering part into the Relief-F algorithm. SFRA can be used in gene selection for microarray gene expression data analysis as the pre-processing step for biomarker detection. The authors believe that SFRA should be versatile enough to be used in various fields that are related to feature selection. However, testing with other datasets is needed to validate this claim.

To further improve the performance of the proposed algorithm, a learning algorithm such as ANN may be introduced to Relief algorithm to enable the adjustment of the weight of features heuristically. This improvement may help the algorithm to avoid local optimums and produce more optimal results. Other unsupervised learning methods than SOM may also be used for comparison purposes. Besides, a systematic development of an adaptive model that is capable of dealing with incremental data is also a goal in a future study.

Acknowledgments. The authors gratefully acknowledge Assoc. Prof. Dr. Nattiya Hirankarn, Lupus Research Unit, Chulalongkorn Hospital, Bangkok, Thailand for providing the expression microarray datasets of hepatitis B patients used in this study.

References

1. Liu, H., Motoda, H.: Computational Methods of Feature Selection. Chapman & Hall/CRC, USA (2008)
2. Raman, B., Ioerger, T.R.: Instance Based Filter for Feature Selection. Journal of Machine Learning Research 1, 1–23 (2002)

3. Liu, B., Cui, Q., Jiang, T., Ma, S.: A Combinational Feature Selection and Ensemble Neural Network Method for Classification of Gene Expression data. BMC Bioinformatics 5 (2004)
4. Cho, S.B., Won, H.H.: Cancer Classification Using Ensemble of Neural Networks with Multiple Significant Gene Subsets. Appl. Intell. 26, 243–250 (2007)
5. Liu, H., Li, J., Wong, L.: A Comparative Study on Feature Selection and Classification Methods Using Gene Expression Profiles and Proteomic Patterns. Genome Informatics 13, 51–60 (2002)
6. Li, T., Zhang, C., Ogihara, M.: A Comparative Study of Feature Selection and Multiclass Classification Methods for Tissue Classification Based on Gene Expression. Bioinformatics 20, 2429–2437 (2004)
7. Liu, C.C., Chen, W.S.E., Lin, C.C., Liu, H.C., Chen, H.Y., Yang, P.C., Chang, P.C., Chen, J.: Topology-based Cancer Classification and Related Pathway Mining Using Microarray Data. Nucleic Acids Res. 34, 4069–4080 (2006)
8. Liu, J., Ranka, S., Kahveci, T.: Classification and Feature Selection Algorithms for Multiclass CGH data. Bioinformatics 24, i86–i95 (2008)
9. Liu, Y.: A Comparative Study on Feature Selection Methods for Drug Discovery. J. Chem. Inf. Comput. Sci. 44, 1823–1828 (2004)
10. Filzmoser, P., Maronna, R., Werner, M.: Outlier Identification in High Dimensions. Computational Statistics & Data Analysis 52, 1694–1711 (2008)
11. Lyons-Weiler, J., Patel, S., Becich, M.J., Godfrey, T.E.: Tests for Finding Complex Patterns of Differential Expression in Cancers: Towards Individualized medicine. BMC Bioinformatics 5 (2004)
12. Zeng, X.Q., Li, G.Z., Yang, J.Y., Yang, M.Q., Wu, G.F.: Dimension Reduction with Redundant Gene Elimination for Tumor Classification. BMC Bioinformatics 9 (2007)
13. Tomlins, S.A., Rhodes, D.R., Perner, S., Dhanasekaran, S.M., Mehra, R., Sun, X.W., Varambally, S., Cao, X., Tchinda, J., Kuefer, R., Lee, C., Montie, J.E., Shah, R.B., Pienta, K.J., Rubin, M.A., Chinnaiyan, A.M.: Recurrent Fusion of *TMPRSS2* and ETS Transcription Factor Genes in Prostate Cancer. Science 310, 644–648 (2005)
14. Tibshirani, R., Hastie, T.: Outlier Sums for Differential Gene Expression Analysis. Biostatistics 8, 2–8 (2006)
15. Sun, Y.: Iterative RELIEF for Feature Weighting: Algorithms, Theories, and Applications. IEEE Transactions on Pattern Analysis and Machine Intelligent 29, 1035–1051 (2007)
16. Robnik-Šikonja, M., Kononenko, I.: Theoretical and Empirical Analysis of ReliefF and RReliefF. Machine Learning 53, 23–69 (2003)
17. Park, H., Kwon, H.C.: Extended Relief Algorithms in Sample-Based Feature Filtering. In: Sixth International Conference on Advanced Language Processing and Web Information Technology, pp. 123–128 (2007)
18. Yin, L., Huang, C.H., Ni, J.: Clustering of Gene Expression Data: Performance and Similarity Analysis. BMC Bioinformatics 7 (2006)
19. Amjady, N., Daraeepour, A.: Day-ahead Electricity Price Forecasting Using the Relief Algorithm and Neural Networks, vol. 978, pp. 1–7. IEEE, Los Alamitos (2008)

Enhanced Visualization by Combing SOM and Mixture Models

Ryotaro Kamimura

IT Education Center, Tokai University
1117 Kitakaname Hiratsuka Kanagawa 259-1292, Japan
ryo@cc.u-tokai.ac.jp

Abstract. In this paper, we propose a simple but powerful method to visualize connection weights by SOM. The conventional SOM has been well established and extensively used to visualize complex data. There have been a number of methods to visualize final connection weights. However, even sophisticated visualization techniques may be ineffective in dealing with ambiguous connection weights due to the complexity of the data set. To cope with this problem, we retrain a network with connection weights obtained by SOM. At this time, we do not optimize networks in terms of errors but we train networks to enhance the characteristics of connection weights at the price of optimality. This enhancement can be realized by smaller Gaussian width. Though these smaller Gaussian widths are not optimal ones in terms of errors, it may give some insights into the characteristics of connection weights. We applied the method to the famous Iris problem and a classification problem for OECD countries. In both problems, we can obtain U-matrices with more explicit boundaries for easy interpretation.

1 Introduction

In this paper, we propose a new method to enhance visualization performance of SOM by additional learning called *enhancement* learning. The conventional SOM [1], [2] has been now widely accepted as one of the fundamental methods in neural networks. There have been many attempts to identify explicitly objective functions and to reformulate it in a framework of information-theoretic approach, Bayesian approach, statistical mechanical approach and mixture models [3], [4], [5]. [6], [7], [8] to cite a few.

In applications, visualization performance is one of the main properties urgently needed. Thus, we need to explore new approach biased toward improved visualization performance. Many sophisticated visualization techniques for SOM [9], [10], [11] have been developed. However, even those sophisticated visualization techniques are of no use if final connection weights are in essence ambiguous. At this point, we think that for better visualization, an optimization principle in terms of errors between input patterns and connection weights should be weakened for more biased solutions with better visualization performance especially for practical applications.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 268–275, 2009.

In this context, we propose a new method, which is simple but efficient in visualization. We combine the conventional SOM with another component that retrains final connection weights by SOM. We choose a mixture model [12] for retaining the network, because of its simplicity. In the first place, we use the conventional SOM to produce final connection weights. By these connection weights, we compute initial prior and posterior probabilities that are given as initial conditions to the mixture model. In the mixture model, connection weights and the prior probability are updated for convergence. However, the Gaussian width σ is fixed through the entire learning processes. The Gaussian width is determined by the enhancement parameter α where the width σ is determined by the inverse of the parameter α. As the enhancement parameter is increased, more detailed parts can be extracted.

In Section 2, we briefly explain a mixture model and how to compute parameters by the EM algorithm. In Section 3, we present two experimental results. In the first example of the Iris problem and the second example of the OECD countries classification problem, we will show that as the enhancement parameter is increased, more detailed boundaries in U-matrices can be seen. In addition, for a certain enhancement parameter, more explicit classes can be obtained.

2 Theory and Computational Methods

2.1 Mixture Model and EM Algorithm

We consider a network shown in Figure 1 and interpret it as a mixture model[12]. Now, suppose that for the jth competitive unit, the sth input pattern x^s is generated by the following Gaussian function

$$p(x^s \mid j) \approx \exp\left(-\frac{\| x^s - w_j \|^2}{2\sigma^2}\right), \tag{1}$$

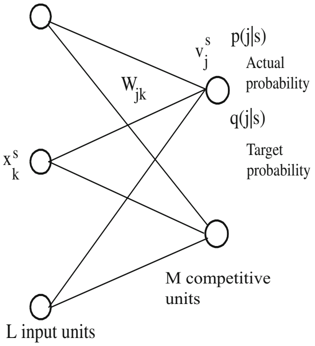


Fig. 1. A network architecture

where w_j denotes connection weights into the jth competitive unit and σ is a Gaussian width. The corresponding posterior probability can be computed by

$$p(j \mid x^s) = \frac{p(x^s \mid j)p(j)}{\sum_{j=1}^{M} p(x^s \mid j)p(j),} \tag{2}$$

where M is the number of competitive units, and $p(j)$ denote a mixing parameter or the prior probability. Then, the negative log-likelihood for the data set or the error function is define by

$$E = -\sum_{s=1}^{S} \log \left\{ \sum_{j=1}^{M} p(x^s \mid j)p(j) \right\}. \tag{3}$$

By the EM algorithm[13], we can directly compute connection weights w_j and the prior probability $p(j)$

$$w_j^{new} = \frac{\sum_{s=1}^{S} p^{old}(j \mid x^s)x^s}{\sum_{s=1}^{S} p^{old}(j \mid x^s)}, \tag{4}$$

and

$$p^{new}(j) = \frac{1}{S} \sum_{s=1}^{S} p^{old}(j \mid x^s), \tag{5}$$

where S is the number of input patterns. For the Gaussian width σ, we do not update it, and then we change it by the following equation

$$\sigma = \frac{1}{\alpha}, \tag{6}$$

where α is an enhancement parameter. As this parameter α is increased, networks are considered to be more enhanced.

2.2 Computational Procedures

In the first place, we apply the SOM to the data set to obtain connection weights w_j and the prior probabilities $p(j)$ and the posterior probability $p(j \mid x^s)$ as shown in Figure 2. Then, we set the enhancement parameter α to be a fixed value ($\alpha > 1$). With these initial conditions, we retrain the network by using a mixture model with the EM algorithm to update connection weights w_j and the prior probabilities $p(j)$. The EM algorithm stops when $\max(\text{mse}(w_j^{new} - w_j^{old}))$ is less than 0.0001.

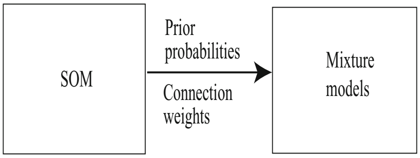
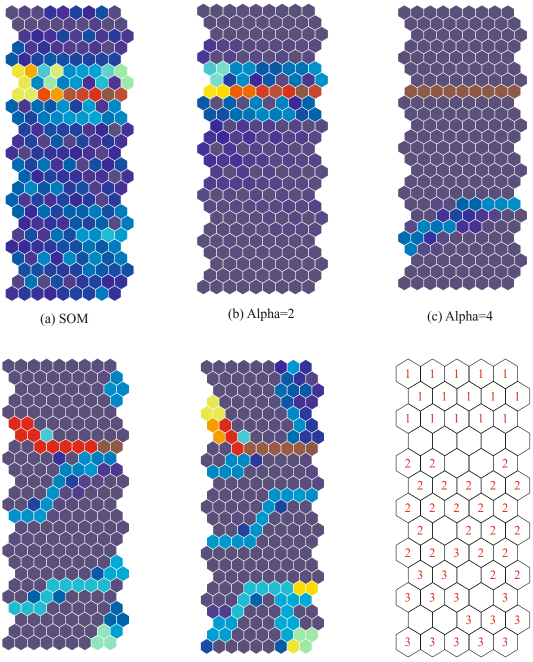


Fig. 2. A computational procedure for enhancement learning.



(a) SOM (b) Alpha=2 (c) Alpha=4

(d) Alpha=6 (e) Alpha=8 (f) Labels

Fig. 3. U-matrix by the conventional SOM (a) and by the enhancement (b)-(e) and labels (f). The enhanced parameter α was increased from 2 (b) to 8 (e). Warmer and cooler colors show larger and smaller values, respectively. Note that labels are the same for all enhancement parameters.

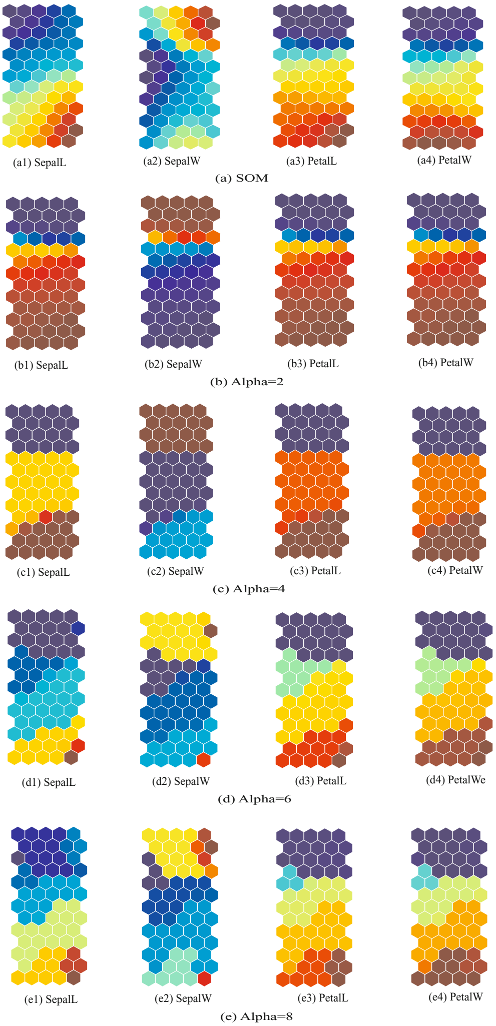
3 Results and Discussion

In the following experiments, we try to show how well the new method extracts features in input patterns. For easy comparison, we use the conventional SOM[1]. All data in the following experiments were normalized and its range is between zero and one.

3.1 Iris Problem

In this experiment, we used the famous Iris problem for easy comparison. Figure 3 shows the U-matrices by SOM(a), the enhancement method (b)-(e) and a map with labels (f). Though the data set is composed of three Iris classes, only two classes can be detected by using the SOM in Figure 3(a). Three classes can be found only by inspecting labels in Figure 3(f). Then, we enhance networks by increasing the enhancement parameter α. When the enhancement parameter α is two, a slightly clearer U-matrix can be obtained but two classes are detected by

[1] We used SOM Toolbox 2.0, February 11th, 2000 by Juha Vesanto http://www.cis.hut.fi/projects/somtoolbox/. No special options were used for easy reproduction.



(a1) SepalL (a2) SepalW (a3) PetalL (a4) PetalW

(a) SOM

(b1) SepalL (b2) SepalW (b3) PetalL (b4) PetalW

(b) Alpha=2

(c1) SepalL (c2) SepalW (c3) PetalL (c4) PetalW

(c) Alpha=4

(d1) SepalL (d2) SepalW (d3) PetalL (d4) PetalWe

(d) Alpha=6

(e1) SepalL (e2) SepalW (e3) PetalL (e4) PetalW

(e) Alpha=8

Fig. 4. Component planes by the conventional SOM (a) and the enhancement method (b)-(e). Warmer and cooler colors show larger and smaller values, respectively.

a boundary in red and brown in Figure 3(b). When the enhancement parameter α is increased to four in Figure 3(c), another boundary in light blue can be found, meaning that the map are divided into three classes. As the enhancement parameter is further increased from six to eight, more detailed classes can be detected in Figure 3(d) and (e).

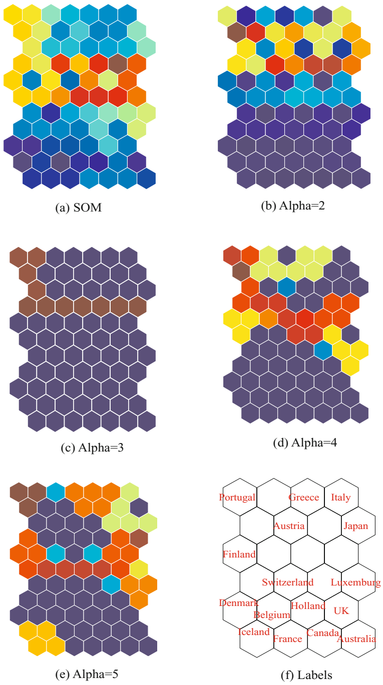
Figure 4 shows component planes by SOM (a) and by the enhancement method (b)-(e). When the enhancement parameter α is two, all component planes represent two parts (Figure 4(b)). When the enhancement parameter α is increased to four in Figure 4(c), all component planes are divided into three parts in terms of the magnitude of connection weights. As the enhancement parameter α is further increased from six to eight, several detailed parts in different colors or magnitudes can be detected in Figure 4(d) and (e).

These experiments clearly show that final U-matrices tend gradually to capture more detailed classification of input patterns as the enhancement parameter σ is increased.

3.2 OECD Classification

We tried to classify 23 OECD countries by using four variables, that is, the total fertility rate, the female labor rate, the tertiary industry labor ratio and the gender development index [14].

Figure 5 shows U-matrices by SOM(a), by the enhancement method (b)-(e) and a map with labels (f). Figure 5(a) shows a U-matrix by the SOM. We can see a relatively wide boundary in red in the middle of the map. The data set seems to be divided into two classes of countries. When the enhancement parameter α is two, strong values on the upper side and lower values on the lower side can be detected. When the parameter is further increased to three in Figure 5(c), a map is completely divided into two parts by a strong boundary in brown colors in the middle. As the parameter is further increased from four to five in Figure 5(d) and (e), additional detailed parts are gradually separated.



(a) SOM

(b) Alpha=2

(c) Alpha=3

(d) Alpha=4

(e) Alpha=5

(f) Labels

Fig. 5. U-matrices by conventional SOM (a), by enhancement (b)-(e) and a map with labels (f). enhanced learning (b). Warmer and cooler colors show larger and smaller values, respectively.

This experimental results also show that as the enhancement parameter is increased, more detailed boundaries can be detected. For a certain value of the enhancement parameter α, the clearest U-matrix can be obtained.

4 Conclusion

In this paper, we have proposed a new method to enhance connection weights obtained by the conventional SOM. The SOM has been now very popular in many applications, because it has strong potentiality to visualize complex data. There have been a number of techniques to visualize connection weights for easy interpretations. However, one of the main problems is that even if sophisticated visualization techniques are used, they are useless for ambiguous connection weights. For making the characteristics of connection weights more explicit, we retrain networks to produce explicit connection weights. In the first place, we train a network with the conventional SOM, and then with the fixed enhancement parameter, the network is retrained with a mixture model with EM algorithm. We applied the method to the famous Iris problem and an OECD countries classification problem. In both problems, we succeeded in making some boundaries by SOM more explicit.

The method here proposed is simple but it gives a very powerful tool to visualize complex data. However, we should note that because we train networks to produce more explicit internal representations at the expense of the errors between input patterns and connection weights, final representations should be interpreted with much reference to original input patterns.

References

1. Kohonen, T.: Self-Organization and Associative Memory. Springer, New York (1988)
2. Kohonen, T.: Self-Organizing Maps. Springer, Heidelberg (1995)
3. Goodhill, G.J., Sejnowski, T.J.: A unifying objective function for topographic mappings. Neural Computation 9, 1291–1303 (1997)
4. Luttrell, S.P.: A Bayesian analysis of self-organising maps. Neural Computation 6(5), 767–794 (1994)
5. Heskes, T.: Self-organizing maps, vector quantization, and mixture modeling. IEEE Transactions on Neural Networks 12(6), 1299–1305 (2001)
6. Bakker, B., Heskes, T.: Clustering ensembles of neural network models. Neural Networks 16, 261–269 (2003)
7. Utsugi, A.: Hyperparameter selection for self-organizing maps. Neural Computation 9(3), 623–635 (1997)
8. Utsugi, A.: Density estimation by mixture models with smoothing priors. Neural Computation 10, 2115–2135 (1998)
9. Kaski, S., Nikkilä, J., Kohonen, T.: Methods for interpreting a self-organized map in data analysis. In: Verleysen, M. (ed.) Proceedings of ESANN 1998, 6th European Symposium on Artificial Neural Networks, Bruges, April 22–24, 1998, pp. 185–190. D-Facto, Brussels, Belgium (1998)

10. Vesanto, J., Alhoniemi, E.: Clustering of the self-organizing map. IEEE-NN 11, 586 (2000)
11. Vesanto, J.: SOM-based data visualization methods. Intelligent-Data-Analysis 3, 111–126 (1999)
12. Bishop, C.M.: Neural networks for pattern recognition. Oxford University Press, Oxford (1995)
13. Dempster, A.P.N., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society (1), 1–38 (1977)
14. Kumagai, E., Funao, N.: Data Mining by R (in Japanese). 9-Ten Publishing Company (2007)

Genetic Versus Nearest-Neighbor Imputation of Missing Attribute Values for RBF Networks

Pedro G. de Oliveira and André L.V. Coelho

Graduate Program in Applied Informatics, University of Fortaleza,
Av. Washington Soares 1321/J30, Fortaleza–CE, Brazil
pedgoncalves@gmail.com, acoelho@unifor.br

Abstract. Missing data is a common issue in almost every real-world dataset. In this work, we investigate the relative merits of applying two imputation schemes for coping with this problem while designing radial basis function network classifiers, which show sensitiveness to the existence of missing values. Whereas the first scheme centers upon the k-nearest neighbor algorithm and has been deployed with success in other supervised/unsupervised learning contexts, the second is based on a simple genetic algorithm model and has not been fully explored so far.

1 Introduction

Real-life datasets invariably contain examples in which the values of some attributes, for some reason or other, are unknown. Aiming at managing this problem, several preprocessing techniques have been devised so far, ranging from those more generic/simple to those more customized/complicated [5]. A broad class of techniques, referred to as imputation methods, involves the replacement of absent values with plausible ones, usually estimated from the data itself. In this context, a common practice is to substitute missing values (MV) with statistical amounts, such as the most likely values, for discrete attributes, or the mean (median) values, for the continuous case.

Since different learning algorithms show different rates of sensitiveness to different levels of data incompleteness, several comparative studies have been conducted recently over different MV handling methods (with an emphasis on those based on imputation) in order to reveal their advantages and disadvantages [1,3,5,9]. In this paper, we move a step further in this initiative and investigate the relative merits of applying two imputation schemes for coping with missing data while specifically designing radial basis function (RBF) networks [6,7]. Whereas the first method centers upon the k-nearest neighbor (KNN) algorithm and has been deployed with success in other supervised/unsupervised learning contexts [3,8], the second is based on a simple genetic algorithm (GA) model [4] and has not been fully explored so far.

In order to contrast, in terms of effectiveness and efficiency, the pros & cons of the genetic and KNN imputation methods for the training and generalization of RBF network classifiers, experiments over UCI benchmark datasets [2] have been

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 276–283, 2009.

conducted, the quantitative results of which are discussed here. In this analysis, we have considered different types of atributes as well as MV rates, and also taken as reference the performance exhibited by mean/mode imputation.

The rest of the paper is organized as follows. Section 2 describes *ein passant* how RBF networks work and comments upon their sensitiveness to MV. The next two sections briefly outline the distinctive aspects behind the nearest-neighbor and genetic imputation methods. Section 5 is dedicated to the analysis of the performance levels exhibited by the contestant imputation schemes, whereas Section 6 concludes the paper and brings remarks on future work.

2 RBF Networks

RBF networks are a popular type of three-layer feedforward networks [6,7]. Each unit i of the hidden layer of this type of network essentially represents a particular point $\mathbf{c}_i$ in the input space, and its output for a given instance $\mathbf{x}$ depends solely on the distance between $\mathbf{c}_i$ and $\mathbf{x}$ (which is achieved by using nonlinear activation functions to convert the distance into a similarity measure). On the other hand, the role of the output layer is to linearly combine the outputs produced at the hidden layer and bring about the actual network's estimates.

An advantage of RBF networks over multilayer perceptrons is that the training of each layer can be conducted separately, bringing efficiency. A disadvantage is that they give every attribute the same weight, meaning that they may show high sensitiveness to the way MV are handled. Although relevant, this issue seems to be overlooked in the literature.

3 Nearest-Neighbor Imputation of Missing Values

This method uses the KNN algorithm [10] to estimate missing data, by considering each time the attribute with MV as class attribute and the others as sources of information for locating the nearest samples. Some advantages of this method include [3]: a) it can estimate both discrete and numeric attribute values; b) it is not necessary to build up a predictive model for each attribute with missing data, as KNN is a lazy-learning method; and c) it can easily treat examples with multiple MV. On ther other hand, efficiency may be a big trouble for this method when considering large datasets if special-purpose data structures (as kD-trees) are not used [10]. Likewise, how to select the value k and the measure of similarity is a decision that may impact the performance results greatly [9].

4 Genetic Imputation of Missing Values

The second scheme employs a customized GA as a mechanism for imputation, which in turn is cast as an optimization problem. So, the values of missing data may be automatically tailored in accord with the supervised learning algorithm being used, leveraging up its performance. Although appealing, to the best of our knowledge, no work has yet fully explored the idea of applying a GA engine to deal specifically with the MV issue. According to the model adopted here, the representation of the individuals is linear, so that each MV is assigned to a position in the string. The encoding can be homogeneous or not, depending on the types of attributes with MV (continuous or discrete). Moreover, fitness-proportionate selection, two-point crossover, and simple mutation are employed for generating novel solutions. Finally, we have made use of the wrapper approach commonly used in the attribute selection task [4] for assigning fitness values. So, these values are set as the average error rate achieved by the RBF network in a stratified 10-fold cross-validation process conducted over the enhanced dataset.

5 Experiments and Results

To assess the performance of the genetic and KNN imputation methods with respect to the training and generalization of RBF network classifiers, extensive experiments have been conducted over UCI benchmark datasets [2].

Due to space limitation, we focus our analysis here on the six datasets depicted in Table 1, which show different number of samples, class distributions, and number/types of attributes and are roughly organized from the less to the more complex ones. These datasets originally do not show absent items, except given to the fourth and sixth in the list. But even in these cases (where we have maintained the MV items), we have followed the same strategy adopted in other papers [3,9], viz., to artificially inject MV completely at random – by this means, the probability of missing data on any attribute would not depend on any value of that attribute.

In the experiments, we have considered three rates of data missingness: 5%, 10%, and 20%. For every original dataset, 10 pairs of training and test (80/20%) partitions were produced via a stratified manner. Then, MV were injected in each novel dataset; this process was repeated five times, giving birth to 50 derived datasets for each original one. In order to assess the performance of the methods, we have recorded the average error rates produced by RBF networks both over the training (10-fold cross-validation) and test datasets. This division is necessary as the genetic imputation uses the cross-validation results as fitness values for its individuals whereas KNN and mean/mode schemes do not.

Table 1. Configuration of the datasets used in the experiments. **NC**: number of classes; **NS**: number of samples; **CD**: class distribution; **NA**: number of attributes; and **TA**: type(s) of attributes (**D**iscrete × **C**ontinuous).

Name	NC	NS	CD	NA	TA
IRIS	3	150	(33.3%;33.3%;33.3%)	4	C
ZOO	7	101	(40.6%;19.8%;4.9%;12.9%;4.00%;7.9%;9.9%)	16	D&C
DIABETES	2	768	(65.1%;34.9%)	8	C
BCD699	2	699	(65.5%;34.5%)	9	C
CAR	4	1,728	(70.0%;22.2%;4.0%;3.8%)	6	D
DERMATOLOGY	6	366	(30.6%;16.7%;19.7%;13.4%;14.2%;5.5%)	33	D&C

Table 2. Average results for 5%-level of missing values

Method	**Crossval**	**Test**	**Generation**	**Time**
	IRIS			
Mean/Mode	92.78 ± 1.00	97.11 ± 3.42	–	0.00 ± 0.10
KNN ($k = 1$)	93.83 ± 1.20	97.33 ± 2.67	–	0.00 ± 0.00
KNN ($k = 3$)	93.94 ± 0.92	97.11 ± 2.69	–	0.00 ± 0.00
KNN ($k = 5$)	94.17 ± 1.17	96.89 ± 3.01	–	0.00 ± 0.00
GA	**96.67 ± 0.60**	**98.28 ± 1.34**	5.07 ± 1.01	96.73 ± 4.40
	ZOO			
Mean/Mode	93.17 ± 1.64	**95.67 ± 4.04**	–	0.00 ± 0.00
KNN ($k = 1$)	95.39 ± 1.22	94.67 ± 3.51	–	0.07 ± 0.12
KNN ($k = 3$)	94.90 ± 2.40	95.33 ± 4.51	–	0.07 ± 0.12
KNN ($k = 5$)	94.57 ± 1.73	95.00 ± 5.00	–	0.13 ± 0.12
GA	**98.19 ± 0.71**	94.29 ± 2.86	3.53 ± 1.42	125.67 ± 9.03
	DIABETES			
Mean/Mode	73.39 ± 0.71	77.82 ± 0.54	–	1.00 ± 0.00
KNN ($k = 1$)	73.03 ± 0.15	**78.87 ± 2.16**	–	1.00 ± 0.00
KNN ($k = 3$)	72.60 ± 0.07	78.47 ± 1.90	–	1.00 ± 0.00
KNN ($k = 5$)	72.70 ± 0.28	78.34 ± 2.41	–	1.00 ± 0.00
GA	**75.63 ± 0.92**	78.36 ± 1.04	8.20 ± 1.06	181.27 ± 4.05
	BCD699			
Mean/Mode	96.12 ± 0.31	**95.92 ± 1.22**	–	1.13 ± 0.23
KNN ($k = 1$)	96.28 ± 0.40	95.39 ± 0.63	–	1.07 ± 0.12
KNN ($k = 3$)	96.20 ± 0.29	95.63 ± 0.98	–	1.00 ± 0.00
KNN ($k = 5$)	96.19 ± 0.42	95.34 ± 0.84	–	1.00 ± 0.00
GA	**97.17 ± 0.41**	95.38 ± 1.15	7.67 ± 0.42	192.13 ± 10.64
	CAR			
Mean/Mode	83.64 ± 0.41	87.38 ± 0.79	–	6.20 ± 0.20
KNN ($k = 1$)	82.69 ± 0.66	86.67 ± 0.99	–	5.00 ± 0.00
KNN ($k = 3$)	83.10 ± 0.53	86.42 ± 0.72	–	5.07 ± 0.12
KNN ($k = 5$)	83.17 ± 0.90	86.77 ± 1.16	–	5.13 ± 0.12
GA	**87.47 ± 0.44**	**87.51 ± 0.29**	8.80 ± 0.60	898.40 ± 79.15
	DERMATOLOGY			
Mean/Mode	95.81 ± 0.72	95.43 ± 1.51	–	4.53 ± 1.42
KNN ($k = 1$)	95.56 ± 0.83	94.89 ± 1.51	–	5.80 ± 1.83
KNN ($k = 3$)	95.84 ± 0.58	95.43 ± 1.98	–	5.20 ± 1.22
KNN ($k = 5$)	95.72 ± 0.74	94.79 ± 1.71	–	5.53 ± 1.63
GA	**97.93 ± 0.45**	**95.77 ± 1.49**	6.33 ± 2.55	596.27 ± 183.15

For the experiments, we have made use of the RBF network model as well as the validation testbench as implemented in the Weka framework [10]. The machine used was an Intel Pentium Core 2, 1.66 GHz equipped with 2 GB of RAM memory. Concerning the values of the control parameters adopted for the GA, they were calibrated as follows: a population of 20 individuals; 10 generations for each execution; 70% as crossover rate; and 10% as mutation rate. This configuration was adopted taking into account the efficiency issue: The GA was

Table 3. Average results for 10%-level of missing values

Method	Crossval	Test	Generation	Time
	IRIS			
Mean/Mode	93.00 ± 1.09	96.44 ± 3.85	–	0.00 ± 0.00
KNN ($k = 1$)	93.50 ± 0.44	96.45 ± 3.42	–	0.00 ± 0.00
KNN ($k = 3$)	93.50 ± 0.50	96.45 ± 3.42	–	0.00 ± 0.00
KNN ($k = 5$)	93.28 ± 0.25	96.22 ± 3.36	–	0.00 ± 0.00
GA	**95.11 ± 0.54**	**98.71 ± 2.24**	6.47 ± 0.23	123.53 ± 1.62
	ZOO			
Mean/Mode	91.52 ± 1.98	**95.67 ± 4.04**	–	0.20 ± 0.20
KNN ($k = 1$)	93.17 ± 1.85	95.33 ± 2.52	–	0.20 ± 0.35
KNN ($k = 3$)	92.59 ± 2.43	93.67 ± 3.06	–	0.47 ± 0.23
KNN ($k = 5$)	93.00 ± 2.95	95.00 ± 2.65	–	0.53 ± 0.12
GA	**97.61 ± 0.38**	95.24 ± 2.86	4.13 ± 1.63	139.07 ± 4.66
	DIABETES			
Mean/Mode	72.48 ± 0.85	77.25 ± 2.35	–	1.00 ± 0.00
KNN ($k = 1$)	72.24 ± 0.13	**78.17 ± 1.93**	–	1.00 ± 0.00
KNN ($k = 3$)	72.25 ± 0.62	77.91 ± 1.71	–	1.00 ± 0.00
KNN ($k = 5$)	72.47 ± 0.75	77.91 ± 1.37	–	1.00 ± 0.00
GA	**74.44 ± 0.34**	76.71 ± 1.04	8.53 ± 1.85	192.67 ± 1.86
	BCD699			
Mean/Mode	96.08 ± 0.26	**95.73 ± 0.79**	–	1.00 ± 0.00
KNN ($k = 1$)	95.90 ± 0.32	95.63 ± 1.34	–	1.00 ± 0.00
KNN ($k = 3$)	96.21 ± 0.16	95.49 ± 1.22	–	1.00 ± 0.00
KNN ($k = 5$)	96.07 ± 0.39	95.54 ± 1.04	–	1.00 ± 0.00
GA	**96.88 ± 0.32**	95.05 ± 1.15	7.20 ± 1.11	199.40 ± 2.95
	CAR			
Mean/Mode	80.11 ± 0.31	85.45 ± 0.29	–	7.40 ± 0.35
KNN ($k = 1$)	78.88 ± 1.03	85.08 ± 0.66	–	5.13 ± 0.23
KNN ($k = 3$)	79.16 ± 0.54	85.41 ± 0.49	–	5.07 ± 0.12
KNN ($k = 5$)	79.43 ± 0.22	**85.94 ± 0.85**	–	5.20 ± 0.20
GA	**84.01 ± 0.39**	85.51 ± 0.58	7.87 ± 0.76	872.00 ± 28.93
	DERMATOLOGY			
Mean/Mode	94.58 ± 1.40	**95.25 ± 0.84**	–	7.07 ± 2.61
KNN ($k = 1$)	94.95 ± 0.82	94.89 ± 2.02	–	8.47 ± 1.45
KNN ($k = 3$)	95.29 ± 1.24	94.52 ± 1.19	–	6.93 ± 0.76
KNN ($k = 5$)	94.74 ± 0.76	94.15 ± 1.94	–	8.20 ± 1.25
GA	**97.47 ± 0.79**	94.68 ± 1.28	6.00 ± 0.35	734.67 ± 120.89

allowed to search for the best MV imputation having available only 200 fitness evaluations in total.

Tables 2–4 bring the results achieved for each combination of dataset, MV rate, and imputation method. In these tables, *Crossval* and *Test* denote the average misclassification rates achieved by the schemes for the training/test data partitions, respectively, whereas *Time* refers to the mean simulation time, in seconds, for a single execution of each imputation technique alone – thus, in case of the GA, it takes into account the time elapsed from the first until the

Table 4. Average results for 20%-level of missing values

Method	**Crossval**	**Test**	**Generation**	**Time**
	IRIS			
Mean/Mode	88.33 ± 1.33	94.22 ± 4.29	–	0.00 ± 0.00
KNN ($k = 1$)	89.39 ± 2.30	96.00 ± 3.71	–	0.07 ± 0.12
KNN ($k = 3$)	89.22 ± 2.12	95.78 ± 4.07	–	0.00 ± 0.00
KNN ($k = 5$)	89.11 ± 2.55	95.78 ± 3.01	–	0.00 ± 0.00
GA	**90.39 ± 0.54**	**98.49 ± 0.37**	6.93 ± 1.86	104.20 ± 1.56
	ZOO			
Mean/Mode	87.57 ± 1.22	94.67 ± 2.89	–	0.73 ± 0.12
KNN ($k = 1$)	87.90 ± 3.48	93.67 ± 3.79	–	1.20 ± 0.35
KNN ($k = 3$)	90.04 ± 0.87	93.67 ± 3.21	–	0.93 ± 0.12
KNN ($k = 5$)	89.63 ± 0.85	94.33 ± 3.06	–	1.00 ± 0.20
GA	**95.23 ± 0.38**	**95.56 ± 2.40**	7.13 ± 1.36	162.60 ± 10.34
	DIABETES			
Mean/Mode	71.58 ± 1.32	77.08 ± 1.32	–	2.20 ± 0.20
KNN ($k = 1$)	70.86 ± 0.73	77.30 ± 1.45	–	2.00 ± 0.00
KNN ($k = 3$)	71.45 ± 0.66	**77.65 ± 1.46**	–	2.00 ± 0.00
KNN ($k = 5$)	71.47 ± 0.75	77.60 ± 1.74	–	2.00 ± 0.00
GA	**72.47 ± 0.25**	73.64 ± 0.39	6.93 ± 1.01	210.53 ± 0.70
	BCD699			
Mean/Mode	95.87 ± 0.32	95.10 ± 0.72	–	2.00 ± 0.00
KNN (K=1)	94.17 ± 0.20	94.96 ± 1.14	–	2.00 ± 0.00
KNN (K=3)	95.48 ± 0.54	**95.39 ± 0.87**	–	2.00 ± 0.00
KNN (K=5)	95.56 ± 0.64	95.39 ± 1.12	–	2.00 ± 0.00
GA	**96.13 ± 0.32**	94.48 ± 1.19	7.93 ± 1.21	207.00 ± 1.40
	CAR			
Mean/Mode	75.13 ± 0.84	**82.75 ± 1.12**	–	9.47 ± 0.46
KNN (K=1)	74.24 ± 0.25	81.99 ± 0.23	–	6.00 ± 0.00
KNN (K=3)	73.82 ± 0.18	81.78 ± 1.87	–	6.00 ± 0.00
KNN (K=5)	74.43 ± 0.23	81.95 ± 1.67	–	6.00 ± 0.00
GA	**77.94 ± 0.24**	82.14 ± 1.33	8.27 ± 0.64	838.60 ± 4.10
	DERMATOLOGY			
Mean/Mode	91.63 ± 0.34	94.61 ± 2.46	–	25.47 ± 9.54
KNN (K=1)	91.15 ± 1.24	94.25 ± 1.45	–	24.33 ± 4.20
KNN (K=3)	91.72 ± 0.84	94.70 ± 1.41	–	23.33 ± 1.14
KNN (K=5)	91.29 ± 0.46	94.25 ± 1.45	–	22.93 ± 3.97
GA	**93.99 ± 0.83**	**94.96 ± 2.17**	6.53 ± 1.90	1,702.47 ± 395.70

last generation of individuals. Conversely, *Generation* indicates the GA iteration at which the best individual was produced for the first time. Highlighted in these tables are the best results achieved (in terms of mean and standard deviation) for both training/test partitions.

From these quantitative results, it is possible to conclude the following. In terms of effectiveness, the three imputation methods were very competitive, with a prevalence of the genetic scheme, when considering the cross-validation results alone, and the mean/mode and genetic schemes, when considering the test results alone. Taking into account both results simultaneously, it is fair to argue that the genetic imputation has outperformed the others in most of the cases, meaning that it could capture better the different forms of attribute interactions. To ratify this perspective, one should note that, as the mean/mode and KNN imputation schemes do not create models of MV data and do not use the cross-validation results for any purpose, their performance over the training dataset should be considered as important as that over the test dataset.

Moreover, as the MV rate increases, the performance of the three methods decreases slightly, with KNN and mean/mode imputation showing higher sensitiveness to this factor for the training partition. Considering the KNN imputation alone, it seems that the choice of k has not influenced so much both in terms of effectiveness and efficiency. For this last criterion, it is worth mentioning that, in these experiments, we have made use of the implementation of the KNN algorithm available in Weka [10], which is very optimized.

Finally, in terms of efficiency, both mean/mode and KNN imputation methods have prevailed significantly over the genetic one, even considering the fact that, in most of the cases, the number of fitness evaluations necessary to locate the final best individual for the first time was lower than the total number of fitness evaluations available. Therefore, one negative aspect of the genetic scheme is its lack of computational scalability, mainly when considering complex/large datasets.

6 Final Remarks

In this work, we have investigated the relative merits of two machine learning based imputation methods for coping with MV while training RBF network classifiers. Whereas the first method centers upon the k-nearest neighbor algorithm, the second is based on a simple genetic algorithm model and has not been fully explored so far. The performance of these schemes were assessed taking into account different types of atributes, MV rates, and UCI benchmark datasets and also considering as yardstick the performance exhibited by the simple mean/mode imputation. Overall, the simulation results indicate that, in terms of effectiveness, the three methods perform comparatively, with the genetic imputation prevailing in terms of higher generalization endowed to the induced RBF networks. However, if efficiency is a requirement, both KNN and mean/mode imputation are more adequate, mainly for complex datasets.

As future work, we plan to assess the impact of choosing other MV imputation methods [1,3,5,9] over the training/generalization of RBF networks as well as to investigate alternative mechanisms to circumvent the scalability problem presented by the genetic imputation method. A hybrid imputation scheme, combining the KNN and genetic ones, is also underway.

References

1. Acuna, E., Rodriguez, C.: The treatment of missing values and its effect in the classifier accuracy. In: Banks, D., et al. (eds.) Classification, Clustering and Data Mining Applications, pp. 639–648. Springer, Heidelberg (2004)
2. Asunción, A., Newman, D.J.: UCI Machine Learning Repository. University of California at Irvine (1998), http://ics.uci.edu/~mlearn/MLRepository.html
3. Batista, G.E., Monard, M.C.: An analysis of four missing data treatment methods for supervised learning. App. Artif. Intel. 17(5), 519–533 (2003)
4. Freitas, A.A.: Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer, Heidelberg (2002)
5. Grzymała-Busse, J.W., Hu, M.: A comparison of several approaches to missing attribute values in data mining. In: Ziarko, W.P., Yao, Y. (eds.) RSCTC 2000. LNCS (LNAI), vol. 2005, pp. 378–385. Springer, Heidelberg (2001)
6. Harpham, C., Dawson, W., Brown, R.: A review of genetic algorithms applied to training radial basis function networks. Neural Comp. & App. 13(3), 193–201 (2004)
7. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice-Hall, Englewood Cliffs (1998)
8. Hruschka, E.R., Hruschka Jr., E.R., Ebecken, N.F.F.: Missing values imputation for a clustering genetic algorithm. In: Wang, L., Chen, K., S. Ong, Y. (eds.) ICNC 2005. LNCS, vol. 3612, pp. 245–254. Springer, Heidelberg (2005)
9. Liu, P., Lei, L.: A review of missing data treatment methods. Int. Journal of Intel. Inf. Manag. Syst. and Tech. 1(3), 412–419 (2005)
10. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)

Combination of Dynamic Reservoir and Feedforward Neural Network for Time Series Forecasting

Štefan Babinec[1] and Jiří Pospíchal[2]

[1] Department of Mathematics, Fac. of Chemical and Food Technologies
Slovak University of Technology, 812 37 Bratislava, Slovakia
Phone/FAX: +421 2 52495177
stefan.babinec@stuba.sk

[2] Institute of Applied Informatics, Fac. of Informatics and Information Technologies
Slovak University of Technology, 842 16 Bratislava, Slovakia
Phone: +421 2 60291548; FAX: +421 2 65420587
pospichal@fiit.stuba.sk

Abstract. Echo State neural networks are a special case of recurrent neural networks. The most important part of Echo State neural networks is so called ”dynamic reservoir”. Echo State neural networks use dynamics of this massive and randomly initialized dynamic reservoir to extract interesting properties of incoming sequences. A standard training of these neural networks uses pseudo inverse matrix for one-step learning of weights from hidden to output neurons. In this approach, we have merged this dynamic reservoir with standard feedforward neural network, with a goal to achieve greater prediction ability. This approach was tested in laser fluctuations and Mackey-Glass time series prediction. The prediction error achieved by this approach was substantially smaller in comparison with prediction error achieved by standard algorithm or time delay neural network with backpropagation algorithm.

1 Introduction

From the point of information transfer during processing, neural networks can be divided into two types: feed-forward neural networks and recurrent neural networks [1]. Unlike the feed forward networks, recurrent neural networks contain at least one cyclical path, where the same input information repeatedly influences the activity of the neurons in a cyclical path. The advantage of such networks is their close correspondence to biological neural networks, but there are many theoretical and practical difficulties connected with their adaptation and implementation. The common problem of all such networks is the lack of an effective supervised training algorithm. This problem was overcome with Echo State neural networks [2]. A very fast algorithm is used in these networks consisting of a calculation of one pseudo-inverse matrix, which is a standard numerical task. But the advantage of ”one step” learning turns into a disadvantage when

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 284–291, 2009.

we try to improve the predictive abilities of the network. The pseudo-inverse matrix approach does not offer any straightforward solution in this case.

On the other hand, classic feedforward neural networks have difficulties with time context. So they cannot be used in standard way for time series forecasting. In this paper we have merged the most important part of Echo State neural network, dynamic reservoir, with standard feedforward neural network. With this approach we lose the advantage of fast computation of the ”one-step” optimization typical for Echo State networks, but we get flexibility and better quality of prediction. Connection between ”liquid state” computing, related to echo states, and backpropagation was mentioned previously in [4]. In our previous work [5] we explored a possibility to improve ”one-step” learning by Anti-Oja's learning.

2 Combination of Dynamic Reservoir and Feedforward Neural Network

Our paper presents a combination of dynamic reservoir and feedforward neural network. In this approach the most complex and at the same time most important part of Echo State neural network, so called ”dynamic reservoir”, remained preserved. The main task of this big recurrent layer is to preprocess the input signal for the feedforward neural network (Fig. 1).

In original Echo State neural networks we have no possibility to stop the training algorithm to avoid the overfitting problem. Therefore such neural networks have often troubles with generalization. We are using backpropagation of error learning algorithm in our approach. So we can stop training when error on validation set does not improve.

Description of neural network. We will mark dynamic reservoir with its input and output neurons as Echo State part of the whole neural network (Fig. 1). This part consists of K input, N hidden and L output neurons. The values of ”input – to – dynamic reservoir” synaptic weights are stored in matrix $\mathbf{W}^{in}$, internal weights are stored in matrix $\mathbf{W}$ and ”dynamic reservoir – to – output” weights are stored in matrix $\mathbf{W}^{out}$. The feedforward neural network consists of L input neurons, M output neurons and of S hidden layers which may have different number of neurons in each layer. As we can see from the Fig.1, the output layer in Echo State part is the same as the input layer in the feedforward neural network.

The learning algorithm. The only weights which are trained in this combination of neural networks are the weights in the feedforward part. The whole algorithm for training and testing consists of two steps.

The first step: The first step should create an untrained Echo State part, consisting of weights $\mathbf{W}^{in}, \mathbf{W}, \mathbf{W}^{out}$, which however can produce so called ”echo” states. There exists a number of ways how to obtain such a network with the given property. We have used the approach presented in [3].

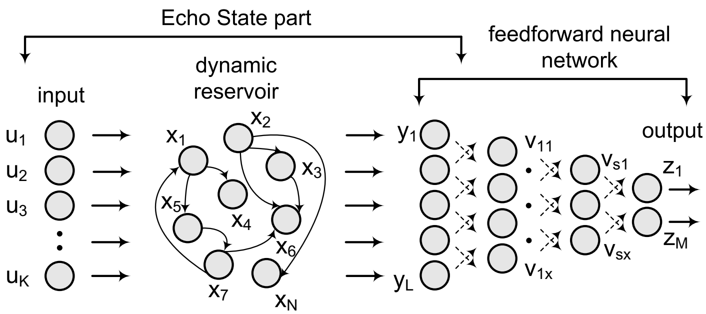


Fig. 1. The architecture used in this approach – combination of dynamic reservoir and feedforward neural network

The second step: Now we can accede to the training and testing of the whole neural network. As we mentioned before, the only weights which are trained in this approach are the weights in the feedforward neural network. The learning algorithm used in this part is the well known backpropagation of error learning algorithm. This algorithm is described in details in [1]. For our approach it is important, how to propagate the input signal through the Echo State part. The states of hidden neurons in dynamic reservoir are calculated from the formula

$$\mathbf{x}(n+1) = f(\mathbf{W}^{in}\mathbf{u}(n) + \mathbf{W}\mathbf{x}(n)), \tag{1}$$

where f is the activation function of hidden neurons (we used the sigmoidal function). The states of output neurons are calculated by the formula

$$\mathbf{y}(n+1) = f^{out}(\mathbf{W}^{out}(\mathbf{u}(n)\mathbf{x}(n+1)), \tag{2}$$

where f^{out} is the activation function of output neurons (we used the sigmoidal function).

3 Comparison of Two Different Approaches

When we look closer at time delay neural network and combination of dynamic reservoir and feedforward neural network (Figure 2), we can see that they are almost identical from architectural point of view. If we stretch dynamic reservoir in one line, we will get a standard input layer almost identical to the input layer used in time delay neural network. The main difference is in input sequence processing. In the case of time delay neural network, the time delayed input signal is introduced to every input neuron in input layer (symbol τ in Figure 2 is time delaying item). Each next neuron in input layer will get input signal, which is time delayed, compared to the previous neuron. On the other hand, in the case of combination of dynamic reservoir and feedforward neural network, every input neuron will receive the same input signal. Perceiving of time context is ensured by recurrent connections in dynamic reservoir and through

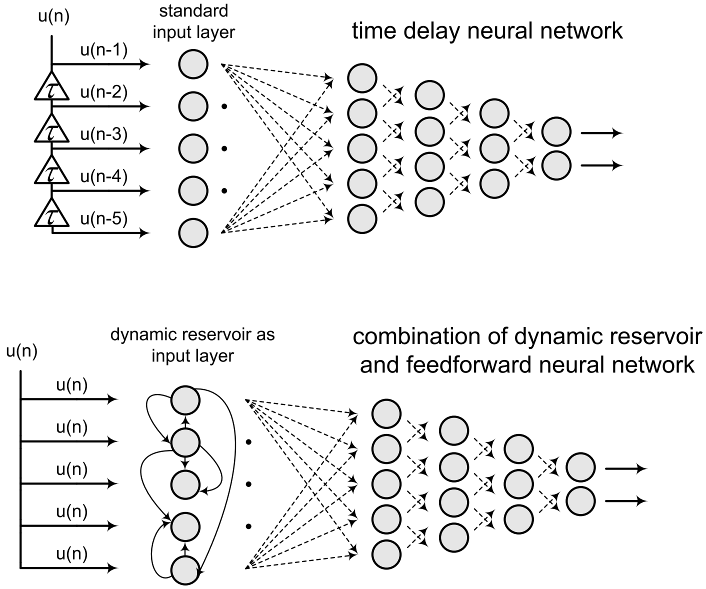

u(n)
standard input layer
time delay neural network
u(n-1)
u(n-2)
u(n-3)
u(n-4)
u(n-5)
τ
u(n)
dynamic reservoir as input layer
combination of dynamic reservoir and feedforward neural network
u(n)
u(n)
u(n)
u(n)
u(n)

Fig. 2. Comparison of two different approaches – time delay neural network and combination of dynamic reservoir and feedforward neural network

the formula applied for calculation of the neurons activities (Equation 1). As the synaptic weights in dynamic reservoir are not adjusted during the training process, it can consist of hundreds of neurons. The training process will be not slowed-down at all.

4 Experiments

We have used two classic benchmarking data sets in this paper. The first testing set was composed of a time sequence of 1000 samples of laser fluctuations data, and the quality of prediction was measured by an error of prediction in the next 100 steps. A mean absolute percentage error (MAPE) was used to measure the quality of prediction of this testing set. The second testing set was composed of a time sequence of 3000 samples of Mackey-Glass (MG) data, and the quality of prediction was measured by an error of prediction in the independently generated 2000 steps. A normalized root mean square error (NRMSE) was used to measure the quality of prediction on this second testing set. The most frequently used prediction horizon in literature is 84 steps ahead in the series. From that reason, we will use the same value. Different error measures were used here to allow eventual comparison with previous results [2]. To conserve the space, the following section provides detailed results of experiments only for laser data set.

The learning was divided into two phases. The first phase was aimed to find the best parameters of dynamic reservoir for quality prediction results from the training set. To reduce computational demands, a very simple feedforward neural

Table 1. Results of representative experiments in the first phase: quality of the laser prediction for different parameter values.

Index	*DR*	*Alpha*	*Average MAPE*	*The best MAPE*
1	200	0.8	33.86 %	31.24 %
2	250	0.8	34.23 %	29.42 %
3	300	0.7	35.94 %	32.85 %
4	350	0.8	38.22 %	34.64 %

network was used in this first phase. The network consisted of two layers with 5 neurons in the first layer and 1 neuron in the second layer. The results for laser data set are in Table 1.

In this table, *DR* represents the dynamic reservoir; *Alpha* is the parameter influencing the ability of the neural network to exhibit echo states. These *DR* and *Alpha* values were chosen in accordance with the proposal used by Jaeger (see [2]). Experiments were carried out in the following way. For each value of *DR* and the parameter *Alpha*, the values of synaptic weights in *DR* were randomly generated 50 times and for each initialization of weights, the error for the testing set was calculated. Further, an average error of all 50 trials is presented in the column *Average MAPE.* Also, the smallest achieved error was recorded in the *Best MAPE* column in the same table. A clear correlation between Best and Average value columns is apparent from Table 1. When a better *Average MAPE* was achieved, there is also a better *Best MAPE.* The best results for laser prediction were achieved with a *DR* consisting of 250 neurons and for the parameter *Alpha* 0.8. For Mackey-Glass prediction, the best results were achieved with a *DR* composed of 400 neurons and for *Alpha* 0.8.

The main experiments were carried out in the second phase with a partial knowledge of behavior of the neural network gained from the first phase. This phase was focused on finding the best parameters of feedfoward neural network and its backpropagation of error learning algorithm. The parameters and initialization of the synaptic weights of the dynamic reservoir were chosen based on the best results from the first phase of the experiments.

The training of synaptic weights in feedforward part using backpropagation of error learning algorithm was carried out for all the samples of the training set. The only exceptions were the last 10 samples in the case of laser prediction. These 10 samples were chosen as the validation set, which was, after the weight adjustment, used for checking the prediction quality of the samples, which were not available for training the neural network. The validation set for Mackey-Glass data was newly generated and was composed of 500 samples. Using this method, a great number of training cycles was carried out, while the use of the validation set monitored the quality of prediction. The representative results for the laser training set are given in Table 2. The variable *Learning cycles* tells how many learning cycles were required to produce the smallest error with the validation set. The variable *Number of neurons* specifies the number of neurons in each layer and *Parameter* γ specifies the value of learning parameter in backpropagation

Table 2. Results of representative experiments for laser testing set in second phase of neural network learning

Index	*Learning cycles*	*Number of neurons*	*Parameter* γ	*MAPE*
1	5108	12 – 1	0.70 %	16.96 %
2	5211	16 – 1	0.80 %	16.24 %
3	9231	16 – 7 – 1	0.85 %	13.84 %
4	7328	12 – 5 – 1	0.85 %	12.92 %

Table 3. Results of three different approaches for laser and Mackey-Glass data sets

Approach	*MAPE*	$NRMSE_{84}$
TD FFNN	21.76 %	0.00051
ESN	29.52 %	0.00034
Combination of dynamic reservoir and FFNN	12.92 %	0.00019

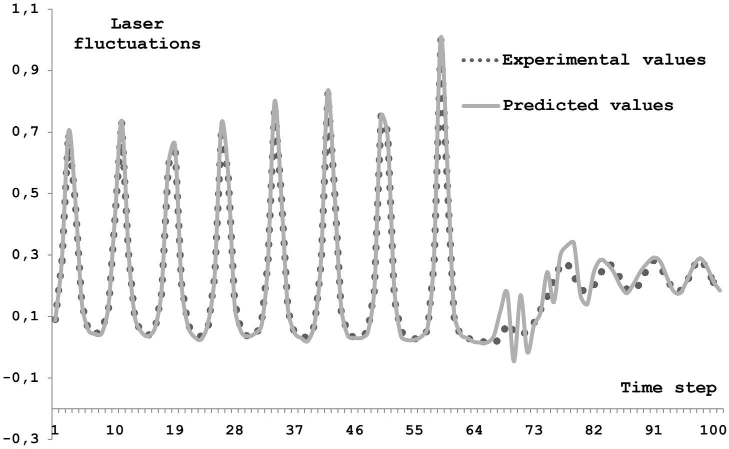


Fig. 3. Testing data: 100 records of laser fluctuations and 100 values predicted by time delay feedforward neural network (MAPE 21.76 %)

of error learning algorithm. Every training of feedforward network started with the same values of synaptic weights and other parameters of dynamic reservoir. Attribute *MAPE* specifies the best reached prediction error on the testing laser set.

In the following Table 3 we can see the comparison of best achieved errors on testing data sets with three different approaches. We can see graphical representation of two approaches in Figures 3 and 4. It is clear from this table and figures that the combination of dynamic reservoir and feedforward neural network can considerably increase the quality of prediction in comparison with

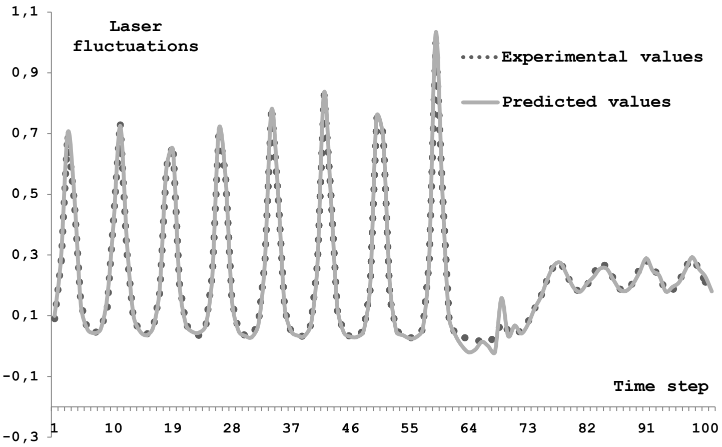


Fig. 4. Testing data: 100 records of laser fluctuations and 100 values predicted by dynamic reservoir combined with feedforward neural network (Experiment No. 4 from Table 2, MAPE 12.92 %)

classic Echo State neural network or time delay feedforward neural network. Attributes *MAPE* and *NRMSE*$_{84}$ in Table 3 specifies the best reached prediction error on the individual testing sets. Attribute *Approach* specifies the approach used for prediction. TD FFNN means time delay feedforward neural network with backpropagation of error learning algorithm. ESN means Echo State neural network with ”one-step” learning algorithm.

5 Conclusions

Echo State neural networks have a substantial advantage over other types of recurrent networks in their ”one-step” learning ability. As a disadvantage, they can be considered to have a relatively low ability to generalize and in general lack an approach, which would be able to improve, at least partially, a previously learned network.

The problem of improving on a previously learned network does not emerge in the case of common feedforward or even other recurrent neural networks. If the need arises to improve the network, a simple adjustment of adding more iterations of the back propagation of error can help. However, this does not work for the Echo State networks, when a standard algorithm allows an ”all or nothing” approach, for example: either we shall teach, or we shall not teach the neural network, but nothing between. The already taught neural network cannot be partially amended by this approach. On the other hand, classic feedforward neural networks are not able to perceive time context. So they cannot be used in standard way for time series forecasting.

The work described in this paper tried to solve both problems, where the main part of Echo State neural network, dynamic reservoir, was combined with feedforward neural network. We have chosen laser fluctuations and Mackey-Glass time series as a testing data. Our aim was to find out if this approach is able to increase prediction quality in comparison with original Echo State neural networks and time delay feedforward neural networks. From the results shown in the paper, it is clear, that this aim has been accomplished. Combination of dynamic reservoir and feedforward neural network can increase the quality of the network's prediction.

Acknowledgement. This work was supported by Scientific Grant Agency Vega of Slovak Republic under grants 1/4053/07, 1/0804/08 and 1/0848/08.

References

1. Haykin, S.: Neural networks - A comprehensive foundation. Macmillan Publishing, New York (1994)
2. Jaeger, H.: The Echo State Approach to Analysing and Training Recurrent Neural Networks. German National Research Center for Information Technology, GMD report 148 (2001)
3. Jaeger, H.: Short Term Memory in Echo State Networks. German National Research Center for Information Technology, GMD report 152 (2002)
4. Natschlager, T., Maass, W., Markram, H.: The “liquid computer”: A novel strategy for real-time computing on time series. Special Issue on Foundations of Information Processing of TELEMATIK 8(1), 39–43 (2002)
5. Babinec, Š., Pospíchal, J.: Improving the Prediction Accuracy of Echo State Neural Networks by Anti-Oja's Learning. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D.P. (eds.) ICANN 2007. LNCS, vol. 4668, pp. 19–28. Springer, Heidelberg (2007)

Learning Nonadjacent Dependencies with a Recurrent Neural Network

Igor Farkaš*

Department of Applied Informatics, Comenius University,
Mlynská dolina, 84248 Bratislava, Slovakia
farkas@fmph.uniba.sk

Abstract. Human learners are known to exploit statistical dependencies of language elements such as syllables or words during acquisition and processing. Recent research suggests that underlying computations relate not only to adjacent but also to nonadjacent elements such as subject/verb agreement or tense marking in English. The latter type of computations is more difficult and appears to work under certain conditions, as formulated by the variability hypothesis. We model this finding using a simple recurrent network and show that higher variability of the intervening syllables facilitates the generalization in the continuous stream of 3-syllable words. We also test the network performance in case of more realistic, two intervening syllables and show that only a more complex training algorithm can lead to satisfactory learning of nonadjacent dependencies.

1 Introduction

Statistical learning appears to be an important mechanism in language development and processing. Humans exploit distributional cues at various levels that help them discover structural dependencies in the language [1,2,3]. These processes are likely to occur unconsiously in the form of implicit learning [4]. In addition to adjacent dependencies, languages tend to comprise relationships between constituents that are conveyed in nonadjacent structure. For example in English, these nonadjacent dependencies exist between subject nouns and verbs in number agreement (e.g. *the boys living next door are naughty*), or between auxiliaries and inflectional morphemes (e.g. *is sleep-ing*). Any mechanism used broadly in language acquisition must therefore, in some way, be capable of learning nonadjacent regularities.

This problem was previously tackled using artificial languages (ALs) and the evidence for tracking nonadjacent probabilities, at least in the continuous streams of syllables, appears contrasting [5,6,7]. Earlier experiments with learning ALs failed to show generalization from statistical information unless additional perceptual cues (i.e. pauses between words or phonological features of phonemes) were available, suggesting that distributional information alone is

* Supported by the grant 1/0361/08 of the Slovak Grant Agency for Science.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 292–299, 2009.

not sufficient to support the discovery of the underlying grammatical-like regularity embedded in a continuous speech stream. With this evidence in mind, Peña et al. [6] argued that generalization and speech segmentation are different processes maintained by separate mechanisms: statistical computations are used in segmentation, but these are distinct from algebraic rule-like computations that would account for generalizations of the distant structure. Peña et al. experimented with learning the continuous stream of 3-syllable words of the form A_iXB_i with $i = 1, 2, 3$ (and three Xs), where A_i exactly predicts B_i. The participants preferred words A_iXB_i, over "part words" (PWs), such as B_jA_iX or XA_iBj (i.e. the triples crossing word boundaries), which was taken as an evidence of successful word segmentation (because the subjects probably took advantage of nonadjacent dependencies between syllables that helped them automatically segment the continuous stream). Next, they were tested whether in addition to segmentation, they could also detect structural regularity in the stream. For that purpose, Peña et al. introduced "rule words" (RWs), such as A_iXB_i, where the intervening (embedded) X appeared in the stream but never in mid-position (i.e. $X \in \{A_j, B_j | j \neq i\}$). This makes RWs congruent with generalization: Unlike PWs, they have a novel surface form (but a familiar deep form). When the subjects' task was to decide between PWs and RWs, no preference for RWs was found, which was interpreted as no generalization (failure to discover the underlying regularity).

However, as promptly suggested by Gómez [8], this could have been due to low variability of X (henceforth, n_X), because she had found that sufficiently large variability ($n_X = 24$) resulted in successful generalization to novel surface structures (RWs). Onnis et al. [9,7] replicated this finding and the results of their experiments led them to fine-tune the variability hypothesis by postulating that generalization occurs at both extremes of variability – zero or large variability. The hypothesis states that when large variability disrupts adjacent dependencies, learners will seek alternative sources of predictability, such as nonadjacent dependencies. In the zero variability case, the reversal effect is observed: the common elements X share the same contextual frames (e.g. *don't-eat-it, he's-eat-ing).* Onnis et al. [7] showed that with sufficiently large n_X, tracking nonadjacent dependencies can result in simultaneous word segmentation and generalization of the embeddings (at the absence of any additional cues). The segmentation of the continuous stream is itself difficult because decreased transitional probabilities (due to high n_X) are known to lead to segmentation within word boundaries [1].

Here we model the effect of variability with a simple recurrent network (SRN; [10] using Peña et al.'s data. SRNs have been successfully applied for various sequential learning tasks, but, to our knowledge, not yet to this type of data with nonadjacent dependencies. In an earlier paper, Garzón [11] used an SRN in this specific task but he did not focus on the variability hypothesis. In experiment 1, we show that generalization accuracy improves with larger variability of embedding. In experiment 2, we simulate the same task in case of more realistic dependencies – embeddings consisting of two syllables rather than one.

2 Simulations

2.1 Experiment 1

Input data. We used streams composed of three different words generated by ALs of the form A_iXB_i, where $P(B_i|A_i) = 1$. In each AL, the three frames $A_i\_B_i$ were combined with embedding X (hence forming various words) whose variability n_X was systematically manipulated. To avoid any biases caused by specific frames, we ran multiple simulations for the same n_X using different frame triples. All three frames had the same probability of occurrence, and so had each X, i.e. $P(X|A_i) = 1/n_X$ and $P(B_i|X) = 0.33$. Hence, all variability conditions had the same transitional probabilities, except $P(X|A_i)$ which depends on n_X. Each syllable was represented as a consonant-vowel pair, taken from the pool of 8 consonants (*b,d,g,p,t,k,r,l*) and 5 vowels (*a,e,i,o,u*), respectively, amounting to 40 possible syllables in total (e.g. *ba, gi, ke*). The 3 frames were randomly chosen with the constraint that no consonant or vowel (except one vowel) was repeated within the same AL (e.g. *da__te, pi__gu, ro__ka*). Words had the embedding formed by syllables that did not occur in the frames (set of size 34). Following Peña et al., PWs had the form B_jA_iX or XB_iA_j. RWs contained embeddings X in A_iXB_i taken from the remaining two frames, i.e. $X \in \{A_j, B_j | j \neq i\}$. This setup allows the following prediction: If a learner computes adjacent statistical probabilities, he should prefer PWs over RWs, at least in the large variability condition (because PWs imply higher transitional probabilities than RWs). Conversely, if the learner computes nonadjacent dependencies he would rely on the most statistically reliable ones, namely $P(B_i|A_i)$, i.e., he would segment correctly at word boundaries, and hence prefer RWs.

Method. We trained an SRN within the next-syllable-prediction paradigm in the stream, given the current syllable at the input. In each simulation, the weights were randomly initialized within [-.1,.1]. Learning rate was set to 0.1 and momentum to 0.8. Each syllable was represented as the concatenation of two localist codes (a consonant and a vowel), to avoid any similarities within consonants or within vowels that might introduce bias into computations. Hence, the network had 13 input and 13 output units. We chose 20 hidden units and 20 context units. In each variability condition (given by n_X), we ran 10 simulations, each using different frames, implying different training and testing sets. For training we used 100 concatenated words (the same words were necessarily repeated within the given set, due to combinatorial limitations), randomly ordered and without pauses. Each simulation lasted 600 epochs. Each testing set contained 12 words. The next syllable (target) was considered to be predicted correctly if the location of both maxima on two output units (one for the consonant and one for the vowel) matched those of the target. To assess the performance of our SRNs, we had to come up with an appropriate procedure that would correspond to the experimental design in Peña et al. In their experiment 2, the subjects were asked to compare RWs and PWs, hearing one after the other, and decide which of the two stimuli sounded more like a word. To match this binary decision task, we compared the prediction errors for both RW

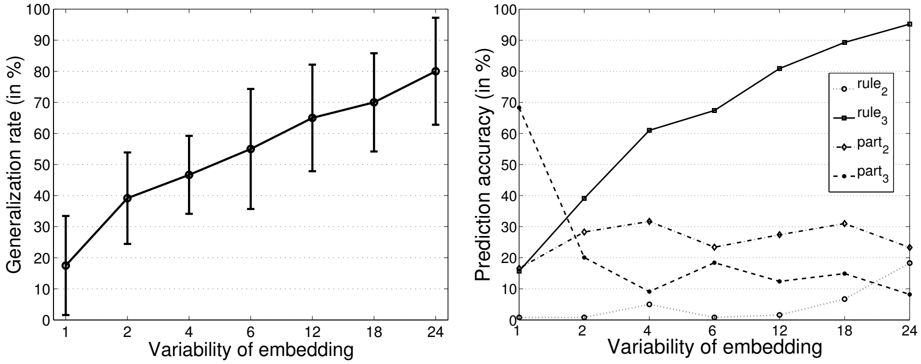


Fig. 1. (a) Average generalization rate in unsegmented artificial languages of type A_iXB_i. (b) Average errors for RWs and PWs, for predicting the X (index 2) and B_i (index 3) syllables. Standard deviations (not shown) were below 20%.

and PW test sets in each simulation as follows: For both test sets we recorded network prediction errors (squared Euclidean distance between the target and the output vectors) in each prediction step. The word prediction error was taken as the sum of prediction errors for the second (X) and the third syllables (B_i). These summed errors for 12 RWs and 12 PWs were then sorted ascendingly. The proportion of RW errors found in the first half of the sorted list was interpreted as the generalization accuracy.

Results. As shown in Figure 1a, the generalization accuracy grows with increasing variability of embedding. When $n_X \geq 12$, the network prefers RWs significantly more often than PWs. Qualitatively, this result is in agreement with experiment 2 in Onnis et al. (2004) although they reported a lower average rate for $n_X = 24$ (64% vs. 80% predicted by our networks). For $n_X = 3$ they reported 42% average generalization rate, which is a very good match with the networks (when considering the average for $n_X = 2$ and 4). We can gain more insight into the model behavior by looking at separate predictions of X and B_i (predictions of the first syllables A_i are not informative and were observed to remain at expected rate 0.33). These SRN predictions for RWs (we will refer to them as RWs of type1) and PWs (syllables X and B) are shown in Figure 1b. It can be seen that whereas predictions within PWs do not improve with higher n_X, predictions of B_i in RWs (denoted as rule$_3$) do significantly. This accounts for preference of RWs over PWs for higher n_X, expressed by lower summed errors in most cases. Similar ascending curve was observed also in case of predicting B_i in RWs which were constructed in a different way (as in Newport & Aslin, 2004) – using novel X syllables that did not appear during training at all (henceforth, RWs of type2). Whichever X is used in RWs, SRN is observed to predict the third RW syllable. Gradual increase of accuracy in predicting B_i in RWs, combined with the previous "input-buffering" step (remembering the first syllable) could be interpreted as the computational implementation of the gradual switching from tracking adjacent to remote dependencies, once the former become less reliable.

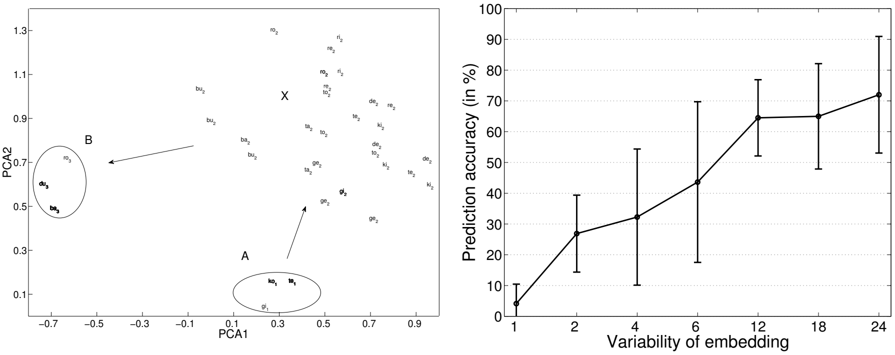


Fig. 2. (a) Layout of hidden unit activations of a trained SRN projected by the principal component analysis method. SRN was trained on language *ko__du*, *gi__ba*, *te__ro* with $n_X = 18$. Z_j denotes the input syllable Z presented at the j-th position within a word. (b) Average prediction accuracy for B_i syllables in rule-words, in unsegmented artificial languages of type A_iXYB_i.

This invariant behavior with respect to RWs of both types can be seen if we look at hidden unit activations of the SRN during testing. Figure 2a shows the two-dimensional (linear) projections of these activation vectors, in case of large variability of embedding ($n_X = 18$). Activations corresponding to A_i syllables are clearly separated, and so are the activations corresponding to B_i syllables. The largest cluster (X) comprises hidden unit activations for intervening inputs, covering syllables used during training (e.g. *ta,de,ki,re*), and also those used in RWs of both type1 (e.g. *ba,ro,gi,te*) and type2 (*ri,to,bu,ge*). Clearly, hidden unit activations document that the first and the last word syllables are distinctly represented in SRN.

However, in case of $n_X < 12$, such a distinction was observed to deteriorate. Although cluster B remained fairly separated, clusters A and X tended to merge, whereas the mutual distance between cluster A and X-B merged cluster became tended to be smaller, too. This may be the reason for lower prediction rates.

Our results do not match the zero part of the variability hypothesis, because preferences for RWs for $n_X = 1$ are very low in Figure 1a. However, according to Onnis et al. [9,7], high RW preference (and hence, generalization) in experiment 1 was only demonstrated in case of segmented artificial speech. If the zero-variability hypothesis turned out to also apply to a continuous stream, it would be a challenge to find a model that could account for that.

2.2 Experiment 2

Input data. To investigate whether an SRN can handle longer dependencies, we created ALs of the type A_iXYB_i, with three different frames per language, and varying embeddings XY within the frames. We considered a simplified design in that for given n_X both X and Y syllables were taken from the same set and varied randomly (i.e. yielding n_X^2 combinations). In this experiment we focused on predictions of RWs which were constructed using existing 3 frames combined with 4 novel embeddings (i.e. 16 possible XY pairs). Out of all possible RWs, we randomly chose 12 of them for testing in each simulation.

Method. Data representation was the same as in experiment 1, as well as the network architecture. However, SRN trained for this task using standard error back-propagation algorithm failed to learn these more distant dependencies. Hence, we used an online version of the real-time recurrent learning (RTRL; [12] which is known to be a more powerful training algorithm for recurrent networks [13]. In this case, SRN was set to have 18 hidden units, was trained for 500 epochs and the learning rate was decreased to 0.05. Other network parameters were the same as in experiment 1.

Results. Figure 2b shows the prediction accuracy for B_i syllables within RWs averaged over 10 simulations. This ability is interpreted as generalization ability for novel words, although predictions of X and Y were very low, inversely related to n_X. Lower prediction accuracy and higher standard deviations compared to previous case (see the rule$_3$ curve in Figure 1b) suggest that this learning task faces greater difficulty.

Peña et al. [6] and Onnis et al. [7] also used two-syllable embeddings in their experiments, but they considered segmented rather than continuous speech. It may be that due to segmentation cue, tested subjects do not find the two-syllable embeddings more difficult in terms of learning generalization. However, our simulations suggest that using two-syllable variable embeddings in case of unsegmented stream does complicate learning. This network prediction could be tested in an experiment with human subjects using unsegmented speech.

3 General Discussion

Statistical learning of dependencies between elements in a sequence is an automatic process widely expoited by humans during processing of temporal structures. Earlier work showed that underlying computations are related to adjacent elements, but more recent work suggests that they also pertain to nonadjacent elements. The latter task appears to be more difficult, perhaps due to learner's bias towards adjacent transitional probabilities that could be perceptually easier to track. In addition, with nonadjacent elements the learner faces a combinatorial problem, since the number of possible nonadjacent probabilities that can be tracked grows exponentially with the length of the embedding. Therefore, it might be that remote computations can only be carried out under certain conditions.

In search for these conditions, earlier research claimed that learning nonadjacent dependencies is only possible given the availability of additional cues. Peña et al.'s conclusion was that pauses between words are necessary, Newport & Aslin [5] stated that phonological cues are required, which explained their finding why only nonadjacent segments could be learnt but not syllables. However, learning nonadjacent dependencies can occur even in a continuous stream of data without any additional cues, provided that the variability of embedding is sufficiently large [7]. Our experiment confirms this computational capability using a sequential learning device, Elman's SRN, that only relies on the order of elements in a sequence. This also implies that the system is capable of focusing on nonadjacent regularities within the frames without having to apply higher algebraic rule-like computations as hypothesized by Peña et al. Actually, the support for ubiqitous associative learning mechanisms was also expressed in the follow-up work that convincingly questioned the line od reasoning used in Peña et al. [14].

In experiment 2 we observed qualitatively the same behavior of SRN in case of longer, two-syllable embeddings, but only if a more powerful RTRL training algorithm was substituted for standard error back-propagation. This more realistic case is very relevant, since natural languages contain remote dependencies, even with typically longer and varying span of embedding (such as A_iXYB_i and A_iXYZB_i). Therefore, suitable experiments and computational simulations should be the focus of subsequent research. In sequence learning literature, earlier work had shown that nonadjacent dependencies spanning identical embedded sequences (of 3 elements and more) are not learnt by human learners and provide an especially difficult learning problem even for large SRNs [15].

On the other hand, tracking remote dependencies requires features reminiscent of learning context-free languages (CFLs). Recurrent neural nets have been shown to have a potential to learn CFLs [16,17]. However, learning processes in these cases are studied on a higher, more abstract level, typically employing only a few symbols (such as the a^nb^n language). This shifts the processing up away from the syllable-based level that involves a considerably higher number of elements.

In summary, tracking remote dependencies is a crucial linguistic ability, whose underpinnings we are just starting to uncover. There are various questions that remain unanswered, one of them being whether adjacent and nonadjacent dependencies require separate learning processes, or the same general process can be employed under a wide range of conditions. Previous results [8,7] and our simulations suggest that the learning system may be capable of various statistical computations seeking the most reliable sources of information. This is consistent with hypotheses of the "reduction of uncertainty" [18] and the simplicity principle [19], stating that the learning system tends to choose the simplest hypothesis about the available data by seeking its invariant patterns. When transitional probabilities are high, adjacent elements are perceived as invariant. When large variability disrupts adjacent probabilities, learners will tune to alternative sources of invariance, potentially between remote elements.

References

1. Saffran, J., Newport, E., Aslin, R.: Word segmentation: The role of distributional cues. Journal of Memory and Language 35, 606–621 (1996)
2. Mintz, T., Newport, E., Bever, T.: The distributional structure of grammatical categories in speech to young children. Cognitive Science 26, 393–424 (2002)
3. Redington, M., Chater, N., Finch, S.: Distributional information: A powerful cue for acquiring syntactic categories. Cognitive Science 22, 425–470 (1998)

4. Cleeremans, A., Destrebecqz, A., Boyer, M.: Implicit learning: news from the front. Trends in Cognitive Sciences 2(10), 406–416 (1998)
5. Newport, E., Aslin, R.: Learning at a distance I. Statistical learning of non-adjacent dependencies. Cognitive Psychology 48, 127–162 (2004)
6. Peña, M., Bonatti, L., Nespor, M., Mehler, J.: Signal-driven computations in speech processing. Science 298, 604–607 (2002)
7. Onnis, L., Monaghan, P., Christiansen, M., Chater, N.: Variability is the spice of learning, and a crucial ingredient for detecting and generalizing in nonadjacent dependencies. In: Proceedings of the 26th Annual Conference of the Cognitive Science Society, pp. 1047–1052. Lawrence Erlbaum, Mahwah (2004)
8. Gómez, R.: Variability and detection of invariant structure. Psychological Science 13(5), 431–436 (2002)
9. Onnis, L., Christiansen, M., Chater, N., Gómez, R.: Reduction of uncertainty in human sequential learning: Evidence from artificial language learning. In: Proceedings of the 25th Annual Conference of the Cognitive Science Society, pp. 886–891. Lawrence Erlbaum, Mahwah (2003)
10. Elman, J.: Finding structure in time. Cognitive Science 14, 179–211 (1990)
11. Garzón, F.: Non-adjacent transitional probabilities and the induction of grammatical reglarities. In: Proceedings of the 27th Annual Conference of the Cognitive Science Society, pp. 767–772. Lawrence Erlbaum Associates, Mahwah (2005)
12. Williams, R., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural Computation 1, 270–280 (1989)
13. Doya, K.: Recurrent networks: Recurrent learning. In: The Handbook of Brain Theory and Neural Networks, pp. 796–800. MIT Press, Cambridge (1995)
14. Perruchet, P., Tyler, M., Galland, N., Peereman, R.: Learning non-adjacent dependencies: No need for algebraic-like computations. Journal of Experimental Psychology: General 133, 573–583 (2004)
15. Cleeremans, A., McClelland, J.: Learning the structure of event sequences. Journal of Experimental Psychology: General 120, 235–253 (1991)
16. Rodriguez, P., Wiles, J., Elman, J.: A recurrent neural network that learns to count. Connection Science 11(1), 5–40 (1999)
17. Bodén, J., Wiles, J.: Context-free and context-sensitive dynamics in recurrent neural networks. Connection Science 12(3/4), 197–210 (2000)
18. Gibson, E.: An Odyssey in Learning and Perception. MIT Press, Cambridge (1991)
19. Chater, N.: Reconciling simplicity and likelihood principles in perceptual organization. Psychological Review 103, 566–581 (1996)

A Back-Propagation Training Method for Multilayer Pulsed Neural Networks Using Principle of Duality

Kaname Iwasa, Mauricio Kugler, Susumu Kuroyanagi, and Akira Iwata

Department of Scientific and Engineering Simulation,
Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya, 466-8555, Japan
kaname@mars.elcom.nitech.ac.jp, mauricio@kugler.com,
{bw,iwata}@nitech.ac.jp

Abstract. Pulsed Neuron (PN) model was proposed as one of the simplest models working by pulse trains. PN model has a membrane potential to deal with the temporal information, and the calculation process is inexpensive. However, as the output function of PN model is an Unit Step function, PN model cannot directly use the back-propagation (BP) method. It would be possible to solve general pattern recognition problems if the PN model could be trained by the BP method. In this paper, we propose a BP method for multilayer pulsed neural networks. The proposed method uses the duality of PN model, in which the desired output of hidden layer neuron is calculated from output layer neurons' weights and output. Experimental results show that the multilayer pulsed neural networks can learn and recognize non-linear problems using the proposed method.

1 Introduction

Artificial neural networks (ANN) are models intended to approximate the way the human brain works. One of the most used methods, the multilayer perceptron (MLP) trained by back-propagation (BP) method was proposed by Rumelhart [1,2], and it is used to solve many pattern recognition problems such as voice recognition.

A possible approach for processing temporal data is the use of ANNs based on Pulsed Neuron (PN) models [3]. This type of neuron deals with input signals on the form of pulse trains, using an internal membrane potential as a reference for generating pulses on its output. PN models can directly deal with temporal data and can be efficiently implemented in hardware, due to its simple structure. Furthermore, high processing speeds can be achieved, as PN model based methods are usually highly parallelizable.

From this advantages, we implemented a PN based network on hardware able to perform sound localization and recognition[4]. In [4], the PN models are trained on software, as the used training algorithm required an weight normalization which is and expensive operation to be implemented in hardware. And

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 300–307, 2009.

Simei *et al* [5,6] proposed a unsupervised training method for spiking (pulsed) neuron model. This method includes the evolving process, it is difficult to be implemented in hardware. Given this fact, there is the need for an training algorithm suitable for hardware implementation. The BP method is well know as a very effective supervised method to train neural networks. However, it cannot be used on PN networks because the PN's output is not differentiable. Sander *et al* [7,8] proposed a BP method for spiking (pulsed) neuron model using temporally encodings. This method works but is too complex to be implemented in hardware.

This paper proposes a back-propagation method for the network, named Multilayer Pulsed Network (ML-PN), suitable for PN networks using the duality of this neuron model [9]. By the duality, the neuron's inputs can be changed with its weights. This is equivalent to calculate the desired inputs and to update weights. The desired inputs are training signals for previous neurons, and previous neurons can learned. And it is a important purpose that the proposed method is easily implemented in hardware. Experimental results show that the ML-PN trained by the proposed method can efficiently recognize non-linear problem.

2 Pulsed Neuron Model

When processing time series data (e.g. sound), it is important to consider the time relations and to have computationally inexpensive calculation procedures to enable real-time processing, requirements fulfilled by the PN model.

Figure 1(a) shows the structure of the PN model. When an input pulse $IN_n(t)$ reaches the n^{th} synapse, the local membrane potential $p_n(t)$ is increased by the value of the weight w_n. The local membrane potentials decay exponentially with a time constant τ_n across time. The neuron's output $o(t)$ is given by:

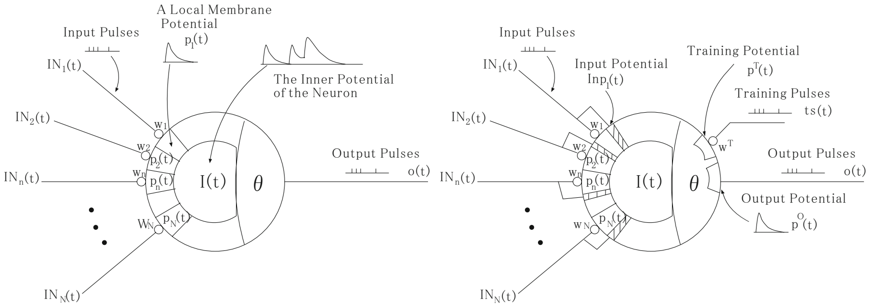
$$o(t) = H(I(t) - \theta) \tag{1}$$

$$I(t) = \sum_{n=1}^{N} p_k(t) \tag{2}$$

$$p_n(t) = w_n IN_n(t) + p_n(t-1)\exp(-\frac{t}{\tau}) \tag{3}$$

where N is the total number of inputs, $I(t)$ is the inner potential, θ is the threshold and $H(\cdot)$ is the unit step function. The PN model also has a refractory period t_{ndti}, during which the neuron is unable to fire, independently of the membrane potential.

For the learning process, we add some components shown in Fig.1(b). $Inp_n(t)$ is a local input potential, p^O is an output potential and p^T is a training potential of this PN. If input pulses are applied to each synapse, the local membrane potential $p_n(t)$ is increased by value of weight w_n, and the local input potential $Inp_n(t)$ is increased by a constant value 1.0. Output potential p^O is increased when the neuron fires. Training potential p^T is increased by training pulses.



(a) Normal model (b) Add the component for Learning

Fig. 1. Pulsed neuron model

3 Proposed Method

3.1 Principle of Duality in Neuron Model

The output of the general neuron model is given by:

$$o = f(\mathbf{w}, \mathbf{i}, \mathbf{\Theta}) \tag{4}$$

where $\mathbf{w}$ is weight vector, $\mathbf{i}$ is input vector, $\mathbf{\Theta}$ is other parameters' vector and $f(\cdot)$ is the output function. The output of neuron is the same to the output of its dual neuron, defined by changing $\mathbf{w}$ and $\mathbf{i}$:

$$o = f(\mathbf{w}, \mathbf{i}, \mathbf{\Theta}) = f(\mathbf{i}, \mathbf{w}, \mathbf{\Theta}) \tag{5}$$

Any neuron which output is obtained by the dot product between inputs and weights presents duality. The PN model also have this property, as shown in Eq.(1)-(3). Thus, it is possible to exchange w_n and IN_n. Equation (6) is the function used to update the weights. By duality it is possible to use the same function to update the desired input values as shown in Eq.(7):

$$\mathbf{w}^{new} = g(\mathbf{w}, \mathbf{i}, \mathbf{\Theta}) \tag{6}$$

$$\mathbf{i}^{new} = g(\mathbf{i}, \mathbf{w}, \mathbf{\Theta}) \tag{7}$$

If updating $\mathbf{w}$ to $\mathbf{w}^{new}$ is used to solve the input pattern problem, updating the $\mathbf{i}$ to $\mathbf{i}^{new}$ can also be used to update the dual neuron for solving the same problem. Thus, the weight update for the dual neuron can be calculated using the same equation used for the normal neuron. The desired input of a neuron is the desired output used for training the previous neuron.

3.2 Neuron Model Learning Rule

Eq.(8) shows the function used to update weight w_{kj}, which connects neuron j from previous layer to neuron k on the current layer, in the supervised learning rule for PN model:

$$w_{kj}(t+1) = w_{kj}(t) + \alpha(p_k^T(t) - p_k^O(t)) \cdot Inp_j(t) \tag{8}$$

where α is the learning rate. The membrane input potential $Inp_j(t)$ is updated by:

$$Inp_j(t) = IN_j(t) + IN_j(t-1) \cdot \exp(-\frac{t}{\tau}) \tag{9}$$

where $IN_j(t)$ is the input pulse from previous neuron j. If τ is constant, $\beta = exp(-\frac{t}{\tau})$ is also constant, thus:

$$Inp_j(t) = IN_j(t) + \beta Inp_j(t-1) = \sum_{a=0}^{T_s} \beta^a IN_j(t-a)$$

and Eq.(8) changes to:

$$w_{kj}(t+1) = w_{kj}(t) + \alpha(p_k^T(t) - p_k^O(t)) \sum_{a=0}^{T_s} \beta^a IN_j(t-a) \tag{10}$$

By duality, changing w to IN gives:

$$T_{kj} = H_j + \alpha(p_k^T(t) - p_k^O(t)) \sum_{a=0}^{T_s} \beta^a w_{kj}(t-a)$$

$$T_j = H_j + \alpha \sum_{k=1}^{K} \{(p_k^T(t) - p_k^O(t)) \sum_{a=0}^{T_s} \beta^a w_{kj}(t-a)\} \tag{11}$$

where H_j is the output from previous neuron j, T_j is the desired output for previous neuron j. The weights of neuron j can be updated by the training signal T_j. Eq.(11) is not completely correct, because of w is update every interaction. However, if the learning rate α is small and $\beta < 1$, Eq.(12) can be approximated to:

$$T_j(t) = H_j + \alpha \sum_{k=1}^{K} \{(p_k^T(t) - p_k^O(t)) w_{kj}(t)\} \equiv H_j + \Delta T_j \tag{12}$$

As T_j is a real number, it is difficult to be treated by a PN model. Therefore, we define a following rule with learning threshold θ_{learn}.

(a) If $|\Delta T_j| < \theta_{learn}$, it means H_j is similar to the desired output T_j. The neuron j should keep this output, $T_j = H_j$.
(b) If $\Delta T_j < -\theta_{learn}$, it means H_j is wrong output for neuron k. The neuron j should not output, $T_j = 0$.
(c) If $\Delta T_j > \theta_{learn}$, it means H_j should be 1. The neuron j should output 1, $T_j = 1$.

This rule is simple to be calculated and can be implemented in hardware. The structure of the learning rule calculating unit is shown in Fig.2.

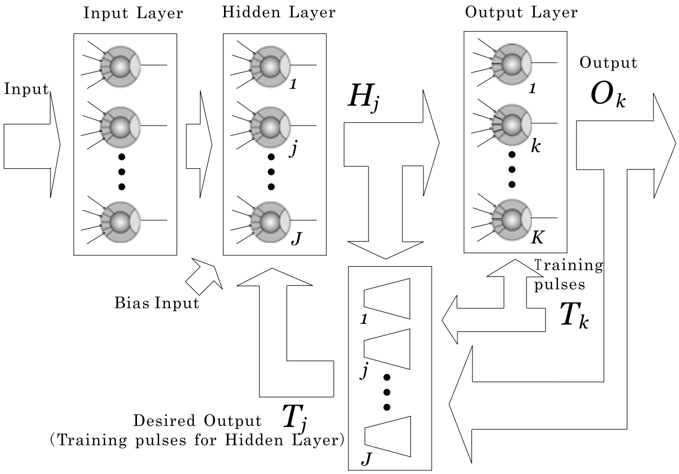


Fig. 2. Structure of Proposed Method

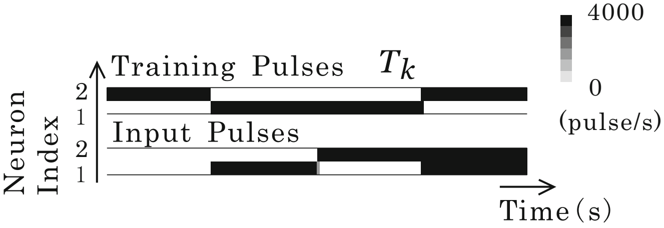


Fig. 3. Input and Training Pulses on XOR Problem

4 Experimental Results

4.1 XOR Problem

At first, the proposed method's efficiency was verified by solving a simple nonlinear problem, an XOR toy-data. Figure 3 shows the input and training pulses, in which the x-axis is time, and y-axis is the neurons' index. The gray level represents the rate of the pulse train, with white corresponding to 0 (no pulse) and black corresponding to 1 (pulse always). Bias pulses are always 1 and are inputted at the same time. This data was used for training ML-PN, according to the parameters shown in Table 1.

Fig. 4 shows the output result after learning. This network output presents the same pulses as the training signal. Hence, the proposed method could successfully learn the XOR problem.

Table2 shows the learning iteration number in each parameters. "×" means it did not converge. When the learning rate α^O is small and learning threshold θ_{learn} is big, this network don't learn. Because of this, these parameters give the effect to training pulses H_j. Hidden layer cannot learn correctly if training pulses H_j don't change.

Table 1. Parameters of ML-PN for XOR Learning

Number of neurons (Input - Hidden - Output)	3 - 4 - 2
Learning Rate α^O / α^M	3.0×10^{-4} / 3.0×10^{-6}
Learning iteration	300
Learning Threshold θ_{learn}	0.5
Sampling Frequency	16 kHz
Input Time Constant τ^{in}	0.005
Output Time Constant τ^O	0.02
Training Time Constant τ^T	0.02
Training Weight Constant w^T	0.063
Refractory Period t_{ndti}	0.001

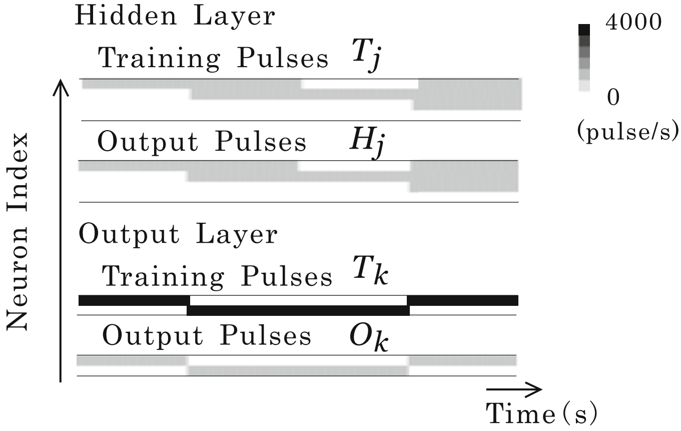


Fig. 4. Output Pulses on XOR Problem

Table 2. The Learning Iterations Number on XOR Problem

	Learning Threshold θ_{learn}			
Learning rate α^O	0.3	0.4	0.5	0.6
2.5×10^{-4}	101	209	×	×
3.0×10^{-4}	136	550	296	×
3.5×10^{-4}	95	106	132	132
4.0×10^{-4}	171	230	130	81

4.2 Sound Recognition Problem

In the next experiment, the proposed method was applied to a real-world problem of sound classification. Six different sound data sets were used on the experiments: “alarm bell”, “interphone”, “kettle” (kettle’s neck sound when boiling water), “phone” (telephone ring), “voice” (one vowel) and “white noise”.

Table 4 shows the accuracy obtained by the proposed method for each learning dataset. The recognition rate is defined as the ratio between the number of

Table 3. Parameters of ML-PN for Sound Learning

Number of neurons (Input - Hidden - Output)	43 - 10 - 6
Learning Rate α^O / α^M	2.0×10^{-4} / 2.0×10^{-6}
Learning Number	700
Learning Threshold θ_{learn}	0.5
Sampling Frequency	16 kHz
Input Time Constant τ^{in}	0.01
Output Time Constant τ^O	0.02
Training Time Constant τ^T	0.02
Training Weight Constant w^T	0.0625
Refractory Period t_{ndti}	0.001

Table 4. The Results of Sound Recognition in Learning Dataset

	Recognition Rate[%]					
Input Sound	alarm bell	interphone	kettle	phone	voice	white noise
alarm bell	100.0	0.0	0.0	0.0	0.0	0.0
interphone	0.0	100.0	0.0	0.0	0.0	0.0
kettle	0.0	0.0	97.7	2.3	0.0	0.0
phone	0.0	0.0	0.0	97.7	2.3	0.0
voice	0.0	0.0	0.0	0.0	100.0	0.0
white noise	0.0	0.0	0.0	0.1	0.0	99.9

neuron's firing corresponding to the sound and the total number of firings. The correct sound source could be recognized with an average accuracy of 99.2%, confirming the efficiency of the ML-PN model for complex real-world data.

5 Conclusions

This paper proposes a back-propagation method for PN model. The proposed method calculates the desired output of the previous neuron from post neuron's weights and error using principle of duality of PN model. The previous neurons learn with the desired outputs. The experimental results confirmed that the ML-PN trained by the proposed method can recognize simple non-linear problems as well as one real-world sound classification problem.

The proposed method consists of simple operations, but there are still multiplication operations in the weights updating process. Future works include changing these multiplications to approximate operations and the implementation of all the learning processes in hardware.

Acknowledgment

This research is supported in part by the Grant-in-Aid for Scientific Research and the Knowledge Clusters (Gifu/Ogaki area), both from the Minister of Education, Culture, Sports, Science and Technology, Government of Japan.

References

1. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning Representations by Backpropagating Errors. Nature 323(9), 533–536 (1986)
2. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning Internal Representation by Error Propagation. In: McClelland, J.L., Rumelhart, D.E., The PDP Research Group (eds.) Parallel Distributed Processing, vol. 1, MIT Press, Cambridge (1986)
3. Maass, W., Bishop, C.M.: Pulsed Neural Networks. MIT Press, Cambridge (1998)
4. Iwasa, K., Kuroyanagi, S., Iwata, A.: A Sound Localization and Recognition System using Pulsed Neural Networks on FPGA. In: Proceeding of International Joint Conference of Neural Networks 2007, August 2007 (to appear)
5. Simei, G.W., Lubica, B., Nikola, K.: Adaptive Learning Procedure for a Network of Spiking Neurons and Visual Pattern Recognition. In: Blanc-Talon, J., Philips, W., Popescu, D., Scheunders, P. (eds.) ACIVS 2006. LNCS, vol. 4179, pp. 1133–1142. Springer, Heidelberg (2006)
6. Simei, G.W., Lubica, B., Nikola, K.: Adaptive Spiking Neural Networks for Audiovisual Pattern Recognition. In: Proceedings of International Conference on Neural Information Processing 2007, pp. 406–415 (2008)
7. Sander, M.B., Han, A.L.P., Joost, N.K.: Error-Backpropagation in Temporally Encoded Networks of Spiking Neurons. In: Proceedings of Neurocomputing, vol. 48, pp. 17–37 (2002)
8. Sander, M.B., Han, A.L.P., Joost, N.K.: Error-Backpropagation in Temporally Encoded Networks of Spiking Neurons, CWI Technical Report, SEN-R0036 (2000)
9. Yamada, K., Kuroyanagi, S., Iwata, A.: A Supervised Learning Method Using Duality in the Artificial Neuron model (Japanese). Proceedings of IEICE J87-D2(2), 399–406 (2004)

Revisiting the Problem of Weight Initialization for Multi-Layer Perceptrons Trained with Back Propagation

Stavros Adam[1], Dimitrios Alexios Karras[2], and Michael N. Vrahatis[3]

[1] Dept. Mathematics, University of Patras Artificial Intelligence Research Center (UPAIRC), GR-26110 Patras, Greece and TEI Hpeirou, Arta, Greece
[2] Dept. Automation , Chalkis Institute of Technology, Psachna, Evoia GR-34400 and Hellenic Open University, Greece
dakarras@{ieee.org,teihal.gr,usa.net}
[3] Dept. Mathematics, University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR-26110 Patras, Greece

Abstract. One of the main reasons for the slow convergence and the suboptimal generalization results of MLP (Multilayer Perceptrons) based on gradient descent training is the lack of a proper initialization of the weights to be adjusted. Even sophisticated learning procedures are not able to compensate for bad initial values of weights, while good initial guess leads to fast convergence and or better generalization capability even with simple gradient-based error minimization techniques. Although initial weight space in MLPs seems so critical there is no study so far of its properties with regards to which regions lead to solutions or failures concerning generalization and convergence in real world problems. There exist only some preliminary studies for toy problems, like XOR. A data mining approach, based on Self Organizing Feature Maps (SOM), is involved in this paper to demonstrate that a complete analysis of the MLP weight space is possible. This is the main novelty of this paper. The conclusions drawn from this novel application of SOM algorithm in MLP analysis extend significantly previous preliminary results in the literature. MLP initialization procedures are overviewed along with all conclusions so far drawn in the literature and an extensive experimental study on more representative tasks, using our data mining approach, reveals important initial weight space properties of MLPs, extending previous knowledge and literature results.

1 Problem Statement and Previous Work

BP training suffers from been very sensitive to initial conditions. In general terms, the choice of the initial weight vector $\boldsymbol{w_0}$ may speed convergence of the learning process towards a global or a local minimum if it happens to be located within the attraction basin of that minimum. Conversely, if $\boldsymbol{w_0}$ starts the search in a relatively flat region of the error surface it will slow down adaptation of the connection weights.

Sensitivity of BP to initial weights, as well as to other learning parameters, was studied experimentally by Kolen and Pollack [1]. Using Monte Carlo simulations on feed forward networks trained with BP to learn the XOR function they discovered that convergence of these networks exhibits a complex fractal-like structure as a function of initial weights. On the other hand, analytical studies for the same problem were

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 308–315, 2009.

reported by Hamey [2] who reconsiders the XOR problem and provides a theoretical study of the error surface for the standard mean square error function. However, he notes the difficulty of having analytic solutions for the general pattern classification case as the study of the error surface is hampered by high dimensionality and because of the difficulty of theoretical analysis. In light of these results it seems that it is not possible, in general, to provide complete theoretical verification for a number of research results claiming to cope effectively with the problem of weight initialization. This is, partially, due to the fact that an exhaustive study of the error surface and of the learning dynamics is almost unfeasible for the general case of the pattern classification problem. On the other hand it is tempting to examine if the initial weight space possesses some kind of structure or if it is able to reveal features which may lead to an effective choice of initial weights. To this end, an effective means seems to be the analysis of the weight space of MLPs in different pattern classification problems. This also permits to gain significant evidence on the validity of different results having either a theoretical basis or proven by experiments.

In this paper we revisit the problem of weight initialization for neural networks trained with gradient descent based procedures. We verify, experimentally, a number of results reported by several researchers for the XOR-network and we extend these results to a well known problem, the IRIS classification problem. Our approach is based on clustering of the weight vectors after having trained an MLP with the BP procedure. Classification of the weight vectors into clusters is performed using unsupervised clustering of Kohonen's self organizing feature maps, or simply self-organizing maps (SOM). Results of our experiments not only reveal, as it was expected, the basins of attraction for the gradient descent learning algorithm, but also provide significant evidence that no inherent clustering exists for the initial weight space. Our approach consists in performing analysis of the weight space after having trained an MLP with the BP procedure for a significant number of weight vectors and for various different sets of training patterns. This approach has already been used by other researchers in the XOR problem, but what is new here is its application to a well known real life problem, the IRIS classification problem. Analysis of the weight space is done using a data clustering and visualization technique. We consider that this approach extends results obtained previously by other researchers. Main considerations of these previous researches are presented hereafter.

2 Analyzing the Weight Space for MLP Using Kohonen's Self Organizing Feature Maps, as a Data Mining Tool for the Analysis

Data clustering and visualization of the clusters, in this paper is based on Kohonen's SOM. The SOM is a type of neural network which is based on unsupervised learning. Thus, unlike supervised learning methods, a SOM is able to perform clustering of data without any reference to the class membership of the input data.

Training the map is an iterative process. At each step a sample vector x is randomly chosen from the input data set and distances between x and all the codebook vectors are computed. Distances between codebook vectors and sample data correspond to similarities between input data and units of the SOM. The best matching unit (BMU), i.e. the most similar unit, is the map unit whose weight vector is closest to x. The training algorithm updates the weight vector of the BMU and of those of its neighborhood so as to get these units move closer to the input vector x, i.e. diminish their distance to the sample vector [3,4,5]. More details on SOM can be found in [3].

The SOM algorithm performs a mapping from the high dimensional input space onto map units. This mapping preserves topology, in the sense that, relative distances between data points in the input space are preserved by distances between map units. This means that data points lying near each other in the input space will be mapped onto neighboring map units. The SOM can thus serve as a clustering tool of high dimensional data. Compared to standard techniques (k-means, ISODATA, competitive learning etc) SOM not only performs better in terms of effectively clustering input data to unknown clusters but also it is computationally more effective [3], [4]. Other comparisons and studies on the data mining capabilities of SOM can be found in the literature. We should mention here the use of the SOM Toolbox for SOM training, data visualization, validation and interpretation. SOM Toolbox was developed at Helsinki University of Technology [5].

We considered two classical benchmarks, the XOR function and the Iris classification problem. The XOR function was studied with a 2-2-1 network while the IRIS classification problem was investigated with two different network architectures, one with 4-10-3 units and another one with 4-5-3 units. For all units the logistic sigmoid was used as an activation function. Experiments for both problems and for different network architectures were carried out according to the following steps:

1. MLPs were trained with the on line BP learning algorithm. All experiments were carried out with the same training parameters, that is interval for initial weights [-2.0, 2.0], learning rate 0.9, max number of epochs 30000 and error between target and actual network output less than 0.01.
2. A relatively large number of weight vectors, that is 5000, were chosen from the initial weight space. Weight vectors were randomly sampled in the interval [-2.0, +2.0] using uniform distribution. After training, the set of weight vectors was roughly divided into two distinct subsets, or categories, of weight vectors. One subset was made up from, those weight vectors for which both, training succeeded (the error goal was reached), and generalization performance was good, i.e. less than 20% of previously unseen patterns rejected per class. These vectors are called the ***successful*** weight vectors while those not meeting the above criteria are called the ***failed*** weight vectors and they fall within the second category.
3. For each weight vector w_i^0 considered before training, the MLP was trained with the on-line gradient descent and a weight vector w_i^* after training was obtained. Thus, gradient descent is considered mapping the weight space before training W onto the weight space after training W'. Given the high dimensionality of these spaces we then used SOMs and projected each one of them on the 2-dimensional space. This approach is graphically depicted in Figure 1.

$$\begin{tikzcd}
W^{(n)} \arrow[r, "gd"] \arrow[d, "som"'] & W'^{(n)} \arrow[d] \\
W^{(2)} & W'^{(2)}
\end{tikzcd}$$

Fig. 1. How SOM could be used as a data mining tool for clustering weight space

4. The 2-dimensional projections of *W* and *W'* thus obtained presented the clusters of weight vectors being discovered by the SOM. Visual inspection of the map representing *W'* permitted to draw some interesting qualitative information regarding the basins of attraction for the gradient descent procedure. Activation of the SOM units and visualization of the unified distance matrix (UM) to identify classification of weight vectors into different clusters. Details on these results are presented in the following section.
5. We, finally, used the possibility offered by the SOM Toolbox to identify the weight vectors for which a unit of the SOM is activated to verify density of *W* regarding convergence and generalization. Actually, given a SOM node in a cluster of successful weight vectors we identified one weight vector before training w_i^0 that gave after training a successful weight vector w_i^*. By injecting additive noise, with normal distribution $N(0,\sigma^2)$, on w_i^0, we took a number of weight vectors in the vicinity of w_i^0. Retraining the MLP with the same BP procedure and mapping the weight vectors after training on the SOM we discovered that even for very small variance many of the noisy weight vectors did not behave the same way as w_i^0.

3 Main Results and Discussion

The tool for presenting results and analyzing them is the unified distance matrix (UM). UM represents the organization of the SOM units into groups, as uniform areas on the 2-dimensional grid.

Result 1. Clustering of the weight vectors after training, which is performed by the SOM without any class membership information, depicts uniform regions of unit activity corresponding to clusters of ***successful*** weight vectors and thin borderline areas for the ***failed*** weight vectors. Figures 2, a and b, visualize clustering of the weights for the 4-10-3 IRIS classification network, while Figures 2, c and d are representative for the 4-5-3 network.

The clusters formed by the SOM correspond to the various minima reached by the gradient descent throughout each experiment. These minima can be global or local. In this sense and together with the topology preservation mapping of the SOM it is straightforward to assume that clusters indicate basins of attraction for the dynamics of the learning procedure. This explains why the number of ***successful*** weight regions for the 4-5-3 IRIS network is less than the respective number for the 4-10-3 IRIS network. This constitutes an experimental confirmation that as the number of unit in the hidden layer increases the number of basins of attraction increases and therefore the study of the weight space becomes more difficult; see Kolen and Pollack [1].

Result 2. Execution of step 5, described above, for a number of different values of σ^2 demonstrated that even for very small variance many of the noisy weight vectors did not behave the same way as the initial vector w_i^0 , i.e. they did not result in successful training. Experiments showed that it is not possible to safely conclude on a minimum "size" for a neighbor of a ***successful*** weight vector in which gradient descent results in ***successful*** weight vectors.

Though important the above results are of practical importance in terms of weight initialization. In order to acquire a better idea on how to deal with this matter we proceeded in a number of experiments using the 4-10-3 MLP for the IRIS problem. During these experiments we used values for the synaptic weight randomly chosen from intervals$\left[-\alpha,+\alpha\right]$, with ***a*** varying from -6.0 up to +6.0, by a step of 0.20. Results of these experiments are stated hereafter.

Result 3. Training seems to be very sensitive to the choice of the training patterns. For the same interval of initial weight vectors and even the same weight vectors, learning curves and subsequent generalization of BP are clearly different.

However, during these trails we did not adopt some specific strategy on how to choose the training patterns and so it remains unclear what characteristic of the input space really biases the learning phase. A possible explanation relies on the inherent structure of the IRIS problem, where two classes are highly correlated. Finally, it seems that a good "strategy" to overcome this problem is to carry out training changing the set of training patterns every 50 or 100 initial weight vectors, these numbers chosen arbitrarily.

Result 4. Training tends to be more successful when the weight vectors are chosen in an interval $[-a,a]$ with $\alpha \approx \sigma_p{}^2$, where $\sigma_p{}^2$ is the maximum variance of the variables of the input pattern space.

While this result is in the same line with some previous research outcomes, it seems that it more accurately reflects a good strategy for weight initialization than previous similar results in the literature. This paper shows that it is not possible to be more specific in the weight initialization range than the above result. More experiments, however, are needed to establish such an outcome.

Result 5. While training seems to be more successful for values of the initial weights within some interval $[-a,a]$ as described above, it is very likely for he BP to give a successful; learning curve for even greater values in intervals $[-ka-a/k,ka]\cup[ka,ka+a/k]$, where k a natural number.

Finally, figures 3,4 below demonstrate the validity of our results 4, 5 above by illustrating how generalization performance is affected by the initialization range when this increases. In the six samples below we see that up to a variance point as indicated by the results 4,5 of the initialization range, there are possibilities for obtaining better generalization than in all other cases. Incrementing this range we find points in the weight space where no solution can be granted, but afterwards, again, there are solutions but with less generalization capability than within the smaller range. This validates the view that even in larger ranges solutions exist, not so successful perhaps, but with less possibility than within the smaller initialization ranges.

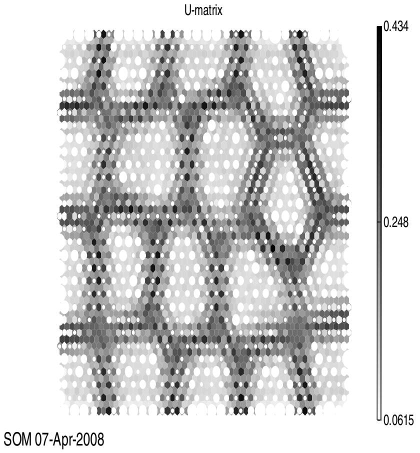


Fig. 2a). Mapping of weight vectors for the Iris network. Mapping of the successful weight vectors for the 10 hidden units Iris network.

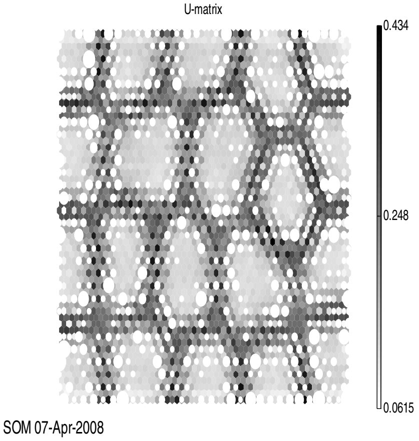


Fig. 2b). Mapping of weight vectors for the Iris network. Mapping of the failed weight vectors for the 10 hidden units the Iris network.

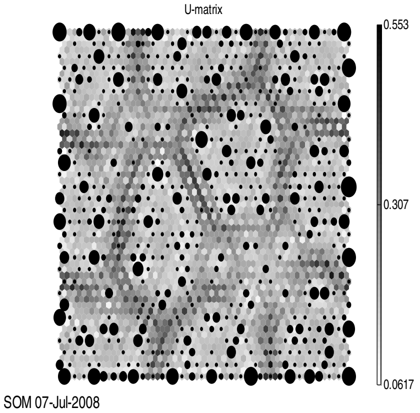


Fig. 2c). Mapping of weight vectors for the Iris network. Mapping of the successful weight vectors for the 5 hidden units Iris network.

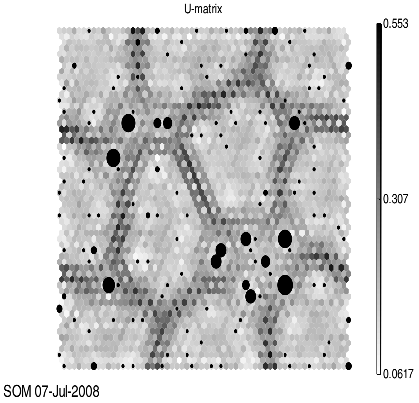


Fig. 2d). Mapping of weight vectors for the Iris network.Mapping of the failed weight vectors for the 5 hidden units Iris network.

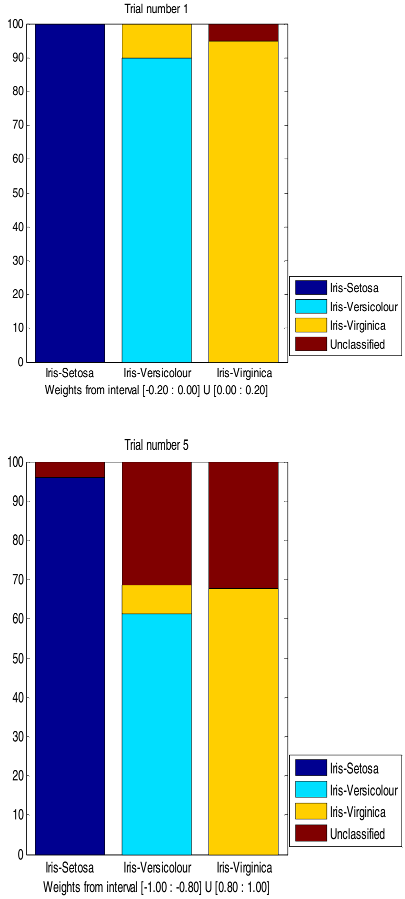


Fig. 3a). How MLP Generalization is affected by initial weights distribution for the Iris network. Misclassification results are shown for selection of the initial weights from the intervals [-0.2 0.0] U [0.0 0.2](upper slide) and [-1 -0.8] U [0.8 1] (lower slide).

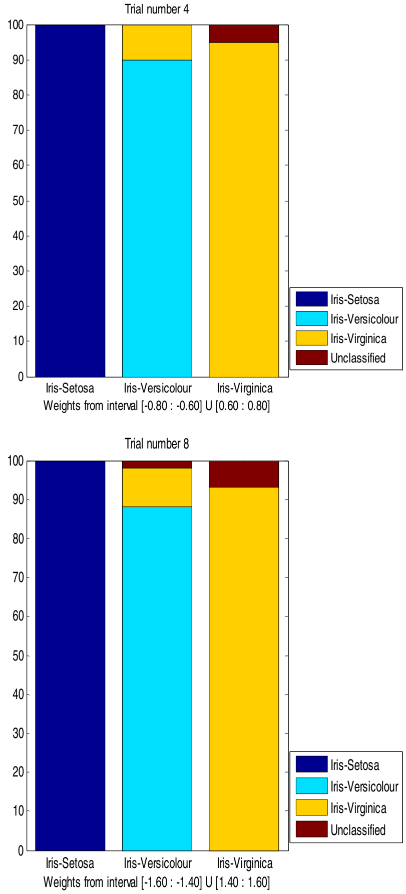


Fig. 3b). How MLP Generalization is affected by initial weights distribution for the Iris network. Misclassification results are shown for selection of the initial weights from the intervals [-0.8 0.6] U [0.6 0.8](upper slide) and [-1.6 -1.4] U [1.4 1.6] (lower slide).

4 Conclusions and Future Trends

This paper revisits MLP initialization problem in the case of BP training and extends literature results both in the description of the weight space as well as in the estimation of a good strategy for selecting weight initialization range. The analysis is performed on a complex classification task, like Iris problem, which is more representative of "real" world problems characteristics than benchmarks used so far in the literature. To this end, a data mining approach, based on Self Organizing Feature Maps (SOM), is involved in this paper. The conclusions drawn from this novel application of SOM algorithm in MLP analysis extend significantly previous preliminary results in the literature. More detailed analysis on real world benchmarks is needed to establish better these results and more elaborate specification of the weight initialization range than the ones of results 4, 5 in this study are needed not, however, too "accurate" as in previous studies. Previous studies have been misleading in this aspect not showing that the weight initialization space is not dense in solutions but it follows an almost fractal structure and, therefore, a probabilistic approach is more suitable in order to find out a good strategy for MLP weight initialization.

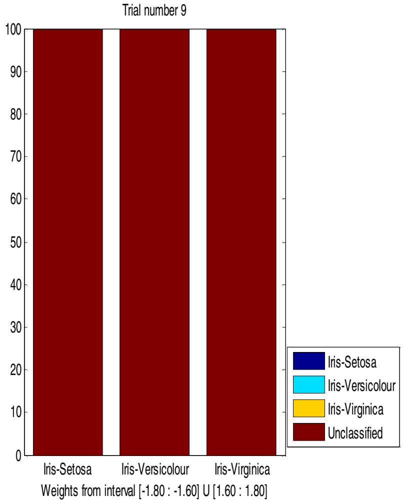


Fig. 4a). How MLP Generalization is affected by initial weights distribution for the Iris network. Misclassification results are shown for selection of the initial weights from the intervals [-1.8 -1.6] U [1.6 1.8].

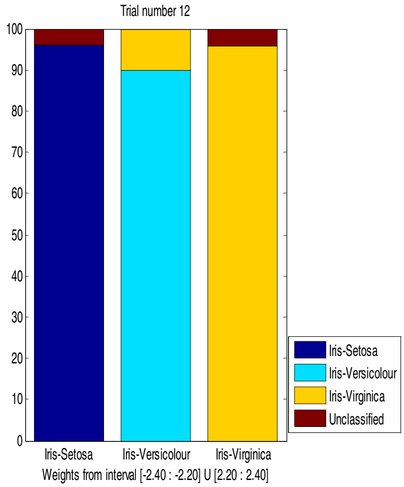


Fig. 4b). How MLP Generalization is affected by initial weights distribution for the Iris network. Misclassification results are shown for selection of the initial weights from the intervals [-2.4 -2.2] U [2.2 2.4].

References

1. Kolen, J.F., Pollack, J.B.: Back propagation is sensitive to initial conditions. In: Advances in Neural Information Processing Systems 3, Denver (1991)
2. Hamey, L.: Analysis of the Error Surface of the XOR Network with Two Hidden Units. In: Proc. 7th Australian Conf. Artificial Neural Networks, pp. 179–183 (1996)
3. Kohonen, T.: Self-Organization and Associative Memory. Springer, Heidelberg (1989)
4. Olli Simula, O., Vesanto, J., Alhoniemi, E., Hollman, J.: Analysis and Modeling of Complex Systems Using the Self-Organizing Map. In: Neuro-Fuzzy Techniques for Intelligent Information Systems (1999)
5. Technical Report on SOM Toolbox 2.0, Helsinki University of Technology (April 2000), `http://www.cis.hut.fi/projects/somtoolbox/`

Analysis on Generalization Error of Faulty RBF Networks with Weight Decay Regularizer

Chi Sing Leung[1], Pui Fai Sum[2], and Hongjiang Wang[1,3]

[1] Dept. of Electronic Engineering, City University of Hong Kong, Hong Kong
[2] Institute of Electronic Commerce, National Chung Hsing University, Taiwan
[3] South China University of Technology, Guangzhou, China

Abstract. In the past two decades, the use of the weight decay regularizer for improving the generalization ability of neural networks has been extensively investigated. However, most existing results focus on the fault-free neural networks only. This papers extends the analysis on the generalization ability for networks with multiplicative weight noise. Our analysis result allows us not only to estimate the generalization ability of a faulty network, but also to select a good model from various settings. Simulated experiments are performed to verify theoretical result.

1 Introduction

The weight decay regularizer for improving the generalization ability has been investigated extensively[1,2]. In this approach, the most important issue is the selection of the regularization parameter. If the value is too small, the trained network well performs on the training set but not on unseen samples. On the other hand, if the value is too large, the trained network cannot well capture the information from the training set. One approach to select the parameter is to use the testing set or cross-validation approaches. We train a number of networks with different regularization parameters. Afterwards, we select the best trained network based on the test set or cross-validation data. However, in many real situations, data is very scarce and we may not have enough data for constructing a test set. Also, the process to test the performance of the trained networks is very time consuming.

Another approach to select the regularization parameter is final prediction error (FPE)[1]. In this approach, we train a number of networks each of which has its own regularization parameter. Afterwards, we predict the generalization ability of these trained networks from the training error based on a so-called prediction error formula. We then select the best trained network based on the estimated FPE.

Although there are a lot of results related to the generalization ability, most of them focus on faulty-free networks only. In the implementation of a neural network, network faults can occur in many different form, such as weight noise[3]. To the best of our knowledge, theoretical result of the generalization error on weight decay trained networks with fault situation has not yet been explored. Hence, it will be useful for estimating the generalization ability for faulty networks.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 316–323, 2009.

In this paper, we will use the radial basis function (RBF) network model as an example to develop the prediction error formula. The background on RBF networks will be presented in Section 2. The fault model is then presented in Section 3. The prediction error formula is then developed in Section 4. Section 5 presents the simulation result. Section 6 concludes our results and discuss the possible extension for the general feedforward network.

2 Data Model and RBF Network with Weight Decay

Throughout the paper, we are given a training dataset,

$$\mathcal{D}_t = \left\{(\mathbf{x}_j, y_j) : \mathbf{x}_j \in \Re^K, y_j \in \Re, j = 1, \cdots, N.\right\},$$

where $\mathbf{x}_j$ and y_j are the input and output samples of an unknown system, respectively. We assume that the dataset $\mathcal{D}_t$ is generated by a stochastic system[2], given by

$$y_j = f(\mathbf{x}_j) + e_j \tag{1}$$

where $f(\cdot)$ is the unknown system mapping, and e_j's are the random measurement noise. The noise terms e_j's are identical independent zero-mean Gaussian random variables with variance equal to S_e. In the RBF approach, we would like to approximate the mapping $f(\cdot)$ by a weighted sum of basis functions, given by

$$f(\mathbf{x}) \approx \hat{f}(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{M} w_i \phi_i(\mathbf{x}), \tag{2}$$

where $\mathbf{w} = [w_1, \cdots, w_M]^T$ is the weight vector, $\phi_i(\cdot)$'s are the radial basis functions given by

$$\phi_i(x) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_i\|^2}{\sigma}\right), \tag{3}$$

the vectors $\mathbf{c}_i$'s are RBF centers, and the positive parameter $\sigma > 0$ controls the width of the RBFs. In the vector-matrix notation, (2) is written as

$$\hat{f}(\mathbf{x}, \mathbf{w}) = \Phi^T(\mathbf{x})\, \mathbf{w}, \tag{4}$$

where $\Phi(\mathbf{x}) = [\phi_1(\mathbf{x}), \cdots, \phi_M(\mathbf{x})]^T$. Our learning task is to find out a weight vector that best fits the observations. Adding a weight decay regularizer [1][4] is a common technique in neural network learning. With a weight decay regularizer, the objective function is given by

$$\mathcal{J}(\mathbf{w}, \lambda) = \frac{1}{N}\sum_{j=1}^{N}(y_j - \Phi^T(\mathbf{x})\, \mathbf{w})^2 + \lambda \mathbf{w}^T \mathbf{w}, \tag{5}$$

where $\lambda \mathbf{w}^T \mathbf{w}$ is the regularizer term, and λ is the weight decay parameter. The weight vector that minimizes the objective function is given by

$$\mathbf{w} = (\mathbf{H}_\phi + \lambda \mathbf{I})^{-1} \frac{1}{N}\sum_{j=1}^{N} \Phi(\mathbf{x}_j) y_j. \tag{6}$$

where $\mathbf{I}$ is the identity matrix, and

$$\mathbf{H}_\phi = \frac{1}{N}\sum_{j=1}^{N} \Phi(\mathbf{x}_j)\Phi^T(\mathbf{x}_j)\,. \tag{7}$$

3 Fault Model

We consider the multiplicative weight noise case [3]. To implement a neural network, the values of the weights must be obtained first. Usually, this is accomplished by running a computer program that executes the training algorithm. Then, the trained weights are encoded using a digital implementation, like on FPGA [5], to realize the neural network. However, this encoding will cause problems as the number representation in FPGA is a low precision floating point format [6], which is different from the format in a computer. The value of a trained weight will have to be rounded to fit the format and then a rounding error will occur. In accordance with the studies in [7], this rounding error is proportional to the magnitude of the number encoded. Therefore, a digital implementation of a computer simulated neural network will lead to a problem identical to adding multiplicative weight noise to that neural network.

In multiplicative weight noise, each implemented weight deviates from its nominal value by a random percent, i.e.,

$$\tilde{w}_{i,b} = w_i + b_i\; w_i \;\; \forall \;\; i = 1, 2, \cdots, M\,, \tag{8}$$

where b_i's are identical independent mean zero random variables with variance S_b. The density function of b_i's are symmetrical. In the matrix-vector form, the output of a faulty network is given by

$$\hat{f}(\mathbf{x}, \tilde{\mathbf{w}}_\mathbf{b}) = \Phi^T(\mathbf{x})\,\tilde{\mathbf{w}}_\mathbf{b}\,. \tag{9}$$

The training error of an implementation $\tilde{\mathbf{w}}_\mathbf{b}$ is then given by

$$\mathcal{E}(\mathcal{D}_t)_\mathbf{b} = \frac{1}{N}\sum_{j=1}^{N}(y_j - \Phi^T(\mathbf{x}_j)\,\tilde{\mathbf{w}}_\mathbf{b})^2\,. \tag{10}$$

From (8), (10) becomes

$$\mathcal{E}(\mathcal{D}_t)_\mathbf{b} = \frac{1}{N}\sum_{j=1}^{N}\Bigg[\left(y_j - \sum_{i=1}^{M}\phi_i(\mathbf{x}_j)w_i\right)^2 + 2\left(\sum_{i=1}^{M}\phi_i(\mathbf{x}_j)b_i w_i\right)\left(y_j - \sum_{i=1}^{M}\phi_i(\mathbf{x}_j)w_i\right) + \sum_{i=1}^{M}\sum_{i'=1}^{M}\phi_i(\mathbf{x}_j)\phi_{i'}(\mathbf{x}_j)b_i b_{i'} w_i w_{i'}\Bigg]\,. \tag{11}$$

Since b_is are identical independent zero mean random variables with symmetric density, the expectation value of $\mathcal{E}(\mathbf{w})_\mathbf{b}$ is equal to

$$\bar{\mathcal{E}}(\mathcal{D}_t)_\mathbf{b} = \frac{1}{N}\sum_{j=1}^{N}\left[\left(y_j - \sum_{i=1}^{M}\phi_i(\mathbf{x}_j)w_i\right)^2 + \sum_{i=1}^{M}S_b\phi_i^2(\mathbf{x}_j)w_i^2\right]\,. \tag{12}$$

$\bar{\mathcal{E}}(\mathcal{D}_t)_{\mathbf{b}}$ is the expected training error over weight noise. Equation (12) can be rewritten in a matrix-vector form:

$$\bar{\mathcal{E}}(\mathcal{D}_t)_{\mathbf{b}} = \mathcal{E}(\mathcal{D}_t) + S_b \mathbf{w}^T \mathbf{G} \mathbf{w} \tag{13}$$

where

$$\mathbf{G} = \mathbf{diag}(\mathbf{H}_\phi) \tag{14}$$

In equation (13), the first term of the right hand side is the training error of a fault-free network while the second term is the error created by the weight noise.

4 Mean Prediction Error

Minimizing the training square error does not mean that the network performs well on an unseen test set. Estimating the generalization performance from the training error is very important. It allows us not only to predict the performance of a trained network but also to select the model from various settings. For the faulty case, we are interesting in estimating the prediction error of the faulty network from a trained network. With this estimation, we no need to use a test set and to generate the possible faulty case to obtain the prediction error.

Let $\mathcal{D}_t = \{(\mathbf{x}_j, y_j)\}_{j=1}^N$ and $\mathcal{D}_f = \{(\mathbf{x}'_j, y'_j)\}_{j=1}^{N'}$, be the training set and the testing set, respectively. For a network with weight noise, the mean training error (MTE) $\bar{\mathcal{E}}(\mathcal{D}_t)_{\mathbf{b}}$ and the mean prediction error (MPE) $\bar{\mathcal{E}}(\mathcal{D}_f)_{\mathbf{b}}$ are given by

$$\bar{\mathcal{E}}(\mathcal{D}_t)_{\mathbf{b}} = \left\langle y^2 \right\rangle_{\mathcal{D}_t} - 2 \left\langle y \varPhi^T(\mathbf{x}) \mathbf{w} \right\rangle_{\mathcal{D}_t} + \mathbf{w}^T \left(\mathbf{H}_\phi + S_b \mathbf{G} \right) \mathbf{w} \tag{15}$$

$$\bar{\mathcal{E}}(\mathcal{D}_f)_{\mathbf{b}} = \left\langle y'^2 \right\rangle_{\mathcal{D}_f} - 2 \left\langle y' \varPhi^T(\mathbf{x}') \mathbf{w} \right\rangle_{\mathcal{D}_f} + \mathbf{w}^T \left(\mathbf{H}'_\phi + S_b \mathbf{G}' \right) \mathbf{w} \,, \tag{16}$$

where $\mathbf{H}_\phi = \frac{1}{N} \sum_{j=1}^N \varPhi(\mathbf{x}_j) \varPhi^T(\mathbf{x}_j)$, $\mathbf{H}'_\phi = \frac{1}{N'} \sum_{j=1}^{N'} \varPhi(\mathbf{x}'_j) \varPhi^T(\mathbf{x}'_j)$, $\mathbf{G} = \mathbf{diag}(\mathbf{H}_\phi)$, $\mathbf{G}' = \mathbf{diag}(\mathbf{H}'_\phi)$, and $\langle \cdot \rangle$ is the expectation operator. Assuming that N and N' are large, $\mathbf{H}'_\phi \approx \mathbf{H}_\phi$, $\mathbf{G}' \approx \mathbf{G}$ and $\left\langle y^2 \right\rangle_{\mathcal{D}_t} \approx \left\langle y'^2 \right\rangle_{\mathcal{D}_f}$. So, the difference between $\bar{\mathcal{E}}(\mathcal{D}_t)_{\mathbf{b}}$ and $\bar{\mathcal{E}}(\mathcal{D}_f)_{\mathbf{b}}$ lies in the difference between their second terms.

We assume that there is an optimal $\mathbf{x}_o$ such that

$$y_j = \varPhi^T(\mathbf{x}_j) \mathbf{w}_o + e_j ; \tag{17}$$

$$y'_j = \varPhi^T(\mathbf{x}'_j) \mathbf{w}_o + e'_j \,, \tag{18}$$

where e_j's and e'_j's are independent zero-mean Gaussian random variables with variance equal to S_e. One should further note that $\mathbf{w}$ is obtained entirely by $\mathcal{D}_t$, which is independent of $\mathcal{D}_f$. Therefore, we can have

$$\left\langle y' \phi^T(x') \mathbf{w} \right\rangle_{\mathcal{D}_f} = \left(\frac{1}{N} \sum_{k=1}^{N'} y'_k \phi^T(x'_k) \right) \mathbf{w} \,. \tag{19}$$

The second term in $\bar{\mathcal{E}}(\mathcal{D}_f)_{\mathbf{b}}$ can thus be given by

$$\begin{aligned} & -2\left\langle y'\Phi^T(\mathbf{x}')\right\rangle_{\mathcal{D}_f}\mathbf{w} \\ & = -2\left(\frac{1}{N}\sum_{j=1}^{N'} y'_j\Phi^T(\mathbf{x'}_k)\right)(\mathbf{H}_\phi+\lambda\mathbf{I})^{-1}\left(\frac{1}{N}\sum_{j=1}^{N} y_j\Phi(\mathbf{x}_j)\right) . \end{aligned} \tag{20}$$

From (17) and (18), the second term in $\bar{\mathcal{E}}(\mathcal{D}_f)_{\mathbf{b}}$ becomes

$$-2\mathbf{w}_o^T\mathbf{H}_\phi\left(\mathbf{H}_\phi+\lambda\mathbf{I}\right)^{-1}\mathbf{H}_\phi\mathbf{w}_o\,.$$

Using a similar method, the second term in $\bar{\mathcal{E}}(\mathcal{D}_t)_{\mathbf{b}}$ is given by

$$-2\frac{S_e}{N}\mathbf{Tr}\left\{\mathbf{H}_\phi\left(\mathbf{H}_\phi+\lambda\mathbf{I}\right)^{-1}\right\}-2\mathbf{w}_o^T\mathbf{H}_\phi\left(\mathbf{H}_\phi+\lambda\mathbf{I}\right)^{-1}\mathbf{H}_\phi\mathbf{w}_o.$$

As a result, the MPE of a faulty RBF network can be in terms of the MTE of the faulty RBF network, given by

$$\bar{\mathcal{E}}(\mathcal{D}_f)_{\mathbf{b}}=\bar{\mathcal{E}}(\mathcal{D}_t)_{\mathbf{b}}+2\frac{S_e}{N}\mathbf{Tr}\left\{\mathbf{H}_\phi\left(\mathbf{H}_\phi+\lambda\mathbf{I}\right)^{-1}\right\}\,. \tag{21}$$

From (13), the MPE of a faulty RBF network can be in terms of the MTE of the fault-free RBF network, given by

$$\bar{\mathcal{E}}(\mathcal{D}_f)_{\mathbf{b}}=\mathcal{E}(\mathcal{D}_t)+2\frac{S_e}{N}\mathbf{Tr}\left\{\mathbf{H}_\phi\left(\mathbf{H}_\phi+\lambda\mathbf{I}\right)^{-1}\right\}+S_b\mathbf{w}^T\mathbf{G}\mathbf{w}. \tag{22}$$

In (22), the term $\mathcal{E}(\mathcal{D}_t)$ is the training error of the trained fault-free network and it can be obtained from the training set. Besides, $\mathbf{H}_\phi$ and $\mathbf{G}$ can also be obtained. The weight noise S_b is assumed to be known. The only unknown variable is the variance of the measurement noise S_e but it can be estimated from the Fedorov's method [8], given by

$$S_e=\frac{1}{N-M}\sum_{j=1}^{N}(y_j-\Phi^T(\mathbf{x}_j)\,\mathbf{H}_\phi^{-1}\frac{1}{N}\sum_{j=1}^{N}\Phi(\mathbf{x}_j)y_j)^2\,. \tag{23}$$

5 Simulations

To verify our result, we consider RBF networks and two problems: (1) function approximation problem and (2) nonlinear time series prediction problem.

The sinc function is a common benchmark [2] and its output is generated by

$$y=\mathrm{sinc}(x)+e, \tag{24}$$

where the noise term e is a mean zero Gaussian noise with variance $\sigma_e^2=0.01$. A training dataset (200 samples) is generated. Also, a testing dataset (1000

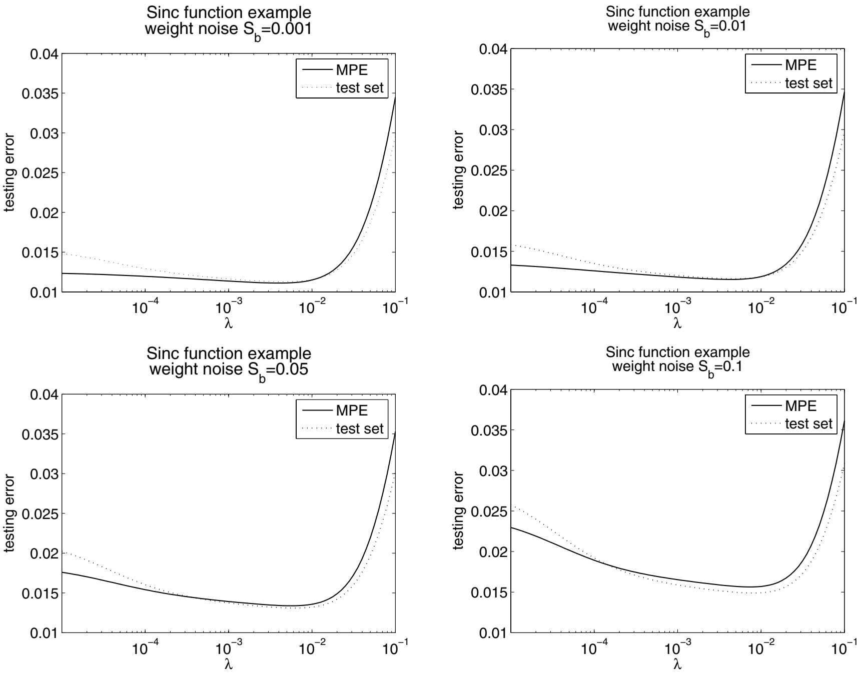


Fig. 1. Prediction error of sinc function example for faulty networks

samples) is generated. The RBF network model has 41 RBF nodes. The 41 centers are selected as $\{-5, -4.75, \cdots, 4.75, 5\}$. The parameter Δ is set to 0.1.

We consider the following nonlinear autoregressive time series[2], given by

$$y(i) = \left(0.8 - 0.5\exp(-y^2(i-1))\right) y(i-1) - \left(0.3 + 0.9\exp(-y^2(i-1))\right) y(i-2) \\ +0.1\sin(\pi y(i-1)) + e(i), \quad (25)$$

where $e(i)$ is a mean zero Gaussian random variable that drives the series. Its variance is equal to 0.09. One thousand samples were generated given $y(0) = y(-1) = 0$. The first 500 data points, were used for training and the other 500 samples were used for testing. Our RBF model is used to predict $y(i)$ based on the past observations, $y(i-1)$ and $y(i-2)$. The prediction is given by

$$\hat{y}(i) = \hat{f}(\mathbf{x}(i), \mathbf{w}) = \sum_{j=1}^{M} w_j \phi_j(\mathbf{x}(i)), \quad (26)$$

where $\mathbf{x}(i) = [y(i-1), y(i-2)]^T$. For this 2D input case, the Chen's LROLS is applied to select important RBF centers (basis functions) from the training samples. The number of selected RBF nodes is 21.

In weight decay, the turning parameter is λ. We illustrate how the MPE can help us to select an appropriate value of λ for minimizing the testing error of

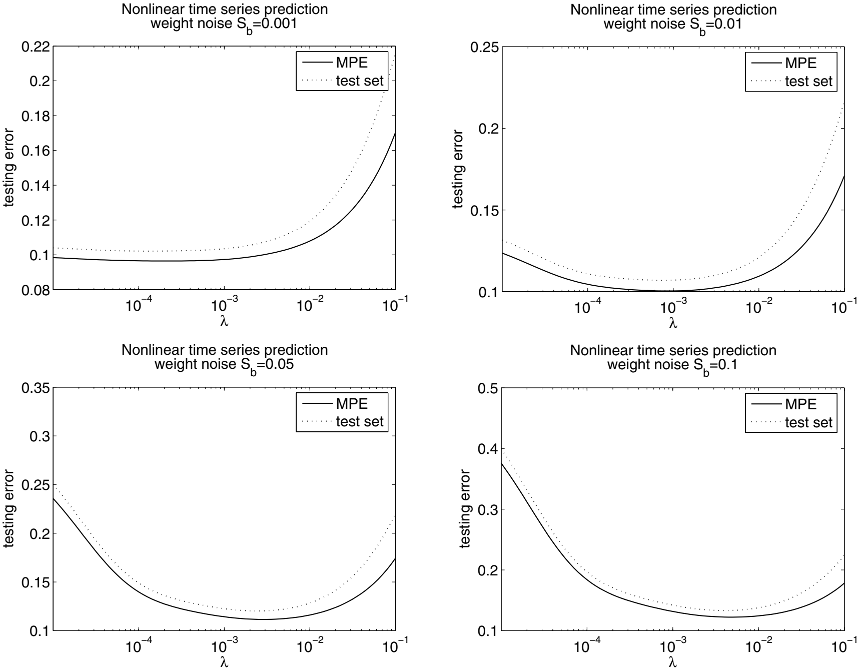


Fig. 2. Prediction error of the nonlinear autoregressive time series prediction for faulty networks

faulty networks. We training a number of networks under different *lambda* values. Afterwards, we calculate the MPE based on our formula. To verify our estimation, we also measure the testing error of faulty networks based on testing sets. For the weight noise fault model, we randomly generate 10, 000 faulty networks for each weight noise level. The results from MPE (obtained from training error of a fault-free network) and the true testing error (from the testing set) are depicted in Figure 1- 2.

From the figures, we can observe that the MPE can accurately locate the appropriate value of λ for minimizing the testing error of faulty networks. For the sinc function example, with weight noise, when $S_b = 0.01$, from the test set evaluation, the λ should be set to around 0.005. With our MPE formula, the optimal λ should be set to 0.0047. When $S_b = 0.1$, from the test set evaluation, the λ should be set to around 0.0078. With our MPE formula, the optimal λ should be set to 0.0075. For the nonlinear autoregressive time series, we obtain the similar result shown in Figure 2.

6 Conclusion

In this paper, the error analysis on the faulty RBF network is presented. Simulation results show that the formulas can help us to select an appropriate value of λ for minimizing the testing error of faulty networks. Although our discussion focuses on RBF networks, one can follow our derivation to handle multilayer networks with other activations, such as sigmoid and hyperbolic tangent. Of course, in such the extended case, we should use some linearization technique to linearize the network function of a multilayer network.

Acknowledgement

The work is supported by the Hong Kong Special Administrative Region RGC Earmarked Grants (Project No. CityU 115606).

References

1. Moody, J.E.: Note on generalization, regularization, and architecture selection in nonlinear learning systems. In: First IEEE-SP Workshop on Neural Networks for Signal Processing, pp. 1–10 (1991)
2. Chen, S., Hong, X., Harris, C.J., Sharkey, P.M.: Sparse modelling using orthogonal forward regression with press statistic and regularization. In: IEEE Trans. Systems, Man and Cybernetics, Part B, pp. 898–911 (2004)
3. Bernier, J.L., Ortega, J., Rodriguez, M.M., Rojas, I., Prieto, A.: An accurate measure for multilayer perceptron tolerance to weight deviations. Neural Processing Letters 10(2), 121–130 (1999)
4. Leung, C.S., Young, G.H., Sum, J., Kan, W.K.: On the regularization of forgetting recursive least square. IEEE Transactions on Neural Networks 10, 1482–1486 (1999)
5. Anitha, D., Himavathi, S., Muthuramalingam, A.: Feedforward neural network implementation in fpga using layer multiplexing for effective resource utilization. IEEE Transactions on Neural Networks 18, 880–888 (2007)
6. Moussa, M., Savich, A.W., Areibi, S.: The impact of arithmetic representation on implementing mlp-bp on fpgas: A study. IEEE Transactions on Neural Networks 18, 240–252 (2007)
7. Kaneko, T., Liu, B.: Effect of coefficient rounding in floating-point digital filters. IEEE Trans. on Aerospace and Electronic Systems AE-7, 995–1003 (1970)
8. Fedorov, V.V.: Theory of optimal experiments. Academic Press, London (1972)

On Node-Fault-Injection Training of an RBF Network

John Sum[1], Chi-sing Leung[2], and Kevin Ho[3,⋆]

[1] Institute of E-Commerce, National Chung Hsing University
Taichung 402, Taiwan
pfsum@nchu.edu.tw

[2] Department of Electronic Engineering, City University of Hong Kong
Kowloon Tong, KLN, Hong Kong
eeleungc@cityu.edu.hk

[3] Department of Computer Science and Communication Engineering,
Providence University, Sha-Lu, Taiwan
ho@pu.edu.tw

Abstract. While injecting fault during training has long been demonstrated as an effective method to improve fault tolerance of a neural network, not much theoretical work has been done to explain these results. In this paper, two different node-fault-injection-based on-line learning algorithms, including (1) injecting multinode fault during training and (2) weight decay with injecting multinode fault, are studied. Their almost sure convergence will be proved and thus their corresponding objective functions are deduced.

1 Introduction

Many methods have been developed throughout the last two decades to improve the fault tolerance of a neural network. Well known methods include injecting random fault during training [20,4], introducing network redundancy [18], applying weight decay learning [7], formulating the training algorithm as a nonlinear constraint optimization problem [8,17], bounding weight magnitude during training [5,12,14], and adding fault tolerant regularizer [2,16,21]. A complete survey on fault tolerant learning methods is exhaustive. Readers please refer to [6] and [23] for reference.

Amongst all, the fault-injection-based on-line learning algorithms are of least theoretical studied. By fault injection, either fault or noise is introduced to a neural network model before each step of training. This fault could either be node fault (stuck-at-zero), weight noise or input noise. As many studies have been reported in the literature on input noise injection [1,3,19,10,11], the primary focus of this paper is on node fault injection. Our companion paper [13] will be focus on weight noise injection.

Suppose a neural network consists of M weights. Let $\theta \in R^M$ be the weight vector of a neural network model and the update equation is given by

[⋆] Corresponding author.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 324–331, 2009.

$\theta(t+1) = \theta(t) - F(x(t+1), y(t+1), \theta(t))$. The idea of node fault injection is to mimic the network that is suffered from random node fault. Before each step of training, each node output is set randomly to either normal or zero (stuck-at-zero). Weight update is then based on this perturbed nodes' output. For simplicity, we let $\tilde{F}(\cdot,\cdot,\cdot)$ be the function corresponding to this perturbed network model. The update equation can readily be defined as follows :

$$\theta(t+1) = \theta(t) - \tilde{F}(x(t+1), y(t+1), \theta(t)). \tag{1}$$

Despite the technique of injecting node fault has appeared for almost two decades [4,7,20], little theoretical analytical result is known about its convergence behavior, the corresponding objective function to be minimized and its extension by adding weight decay during training an RBF network.

In this paper, two node-fault-injection-based on-line learning algorithms, namely (1) injecting multinode fault [4,20] during training and (2) weight decay with injecting multinode fault [7], will be analyzed. Analysis on weight-noise-injection-based training will be presented in another paper. Their corresponding objective functions and their convergence properties will be analyzed analytically. The major technique is by applying the Gladyshev Theorem in the theory of Stochastic Approximation [9]. The definition of a RBF model and the node fault injection training algorithms will be introduced in the next section. Then, the main results on their convergence properties and the objective functions will be stated in Section 3. The proof of theorems will be presented in Section 4. Section 5 will give a conclusion.

2 RBF Training with Node Fault Injection

Let $\mathcal{M}_0$ be an unknown system to be modeled. The input and output of $\mathcal{M}_0$ are denoted by x and y respectively. The only information we know about $\mathcal{M}_0$ is a set of measurement data $\mathcal{D}$, where $\mathcal{D} = \{(x_k, y_k)\}_{k=1}^{N}$. Making use of this data set, an estimated model $\hat{\mathcal{M}}$ that is *good* enough to capture the *general behavior* of the unknown system can be obtained. For $k = 1, 2, \cdots, N$

$$\mathcal{M}_0 \quad : \quad y_k = f(x_k) + e_k, \tag{2}$$

where (x_k, y_k) is the k^{th} input-output pair that is measured from an unknown deterministic system $f(x)$ with random output noise e_k, $e_k \sim \mathcal{N}(0, S_e)$. To model the unknown system, we assume that $f(x)$ can be realized by an RBF network consisting of M hidden nodes, i.e.

$$y_k = \sum_{i=1}^{M} \theta_i^* \phi_i(x_k) + e_k \tag{3}$$

for all $k = 1, 2, \cdots, N$ and $\phi_i(x)$ for all $i = 1, 2, \cdots, M$ are the radial basis functions given by

$$\phi_i(x) = \exp\left(-\frac{(x - c_i)^2}{\sigma}\right), \tag{4}$$

where c_is are the centers of the radial basis functions and the positive parameter $\sigma > 0$ controls the width of the radial basis functions. Thus, a model $\mathcal{M}$ in Ω is represented by an M-vector, $\theta^* = (\theta_1^*, \theta_2^*, \cdots, \theta_M^*)^T$ and the model set Ω will be isomorphic to R^M.

2.1 Multinode Fault Injection Training

In conventional training by minimizing MSE, the update equation for $\theta(t)$ is given by

$$\theta(t+1) = \theta(t) + \mu_t(y_t - \phi^T(x_t)\theta(t))\phi(x_t), \tag{5}$$

where μ_t (for $t \geq 1$) is the step size at the t^{th} iteration. While an RBF network is trained by multinode fault injection, the update equation is given by

$$\theta(t+1) = \theta(t) + \mu_t(y_t - \tilde{\phi}^T(x_t)\theta(t))\tilde{\phi}(x_t), \tag{6}$$

$$\tilde{\phi}_i = (1-\beta_i)\phi_i, \quad P(\beta_i = 1) = p, \ \forall\, i = 1, \cdots, M. \tag{7}$$

We assume that all nodes are of equal fault rate p, i.e.

$$P(\beta_i) = \begin{cases} p & \text{if } \beta_i = 1 \\ 1-p & \text{if } \beta_i = 0. \end{cases} \tag{8}$$

for $i = 1, 2, \cdots, M$, Besides, $\beta_1, \cdots, \beta_M$ are independent random variables.

2.2 Weight Decay-Based Multinode Fault Injection Training

The update equation for weight decay-based multinode fault injection training is similar to that of simple multinode fault injection, except that a decay term is added. For a RBF network, $f(x_t, \theta(t)) = \phi(x_t)^T\theta(t)$, that is trained by injecting multinode fault during weight decay learning,

$$\theta(t+1) = \theta(t) + \mu_t \left\{(y_t - \tilde{\phi}^T(x_t)\theta(t))\tilde{\phi}(x_t) - \lambda\theta(t)\right\}, \tag{9}$$

$$\tilde{\phi}_i = (1-\beta_i)\phi_i, \tag{10}$$

for all $i = 1, \cdots, M$. The definition of the random variable β_i is the same as before. $P(\beta_i) = p$ if $\beta_i = 1$ and $(1-p)$ otherwise.

3 Main Results

Theory of stochastic approximation has been developed for more than half a century for the analysis of recursive algorithms. Advanced theoretical works for complicated recursive algorithms have still been under investigation [15]. The theorem applied in this paper is based on Gladyshev Theorem [9].

Theorem 1 (Gladyshev Theorem [9]). *Let $\theta(t)$ and $M(\theta(t), \omega(t))$ for all $t = 0, 1, 2,$ and so on be m-vectors. $\omega(t)$ for all $t = 0, 1, 2,$ and so on are i.i.d. random vectors with probability density function $P(\omega)$ [1]. Consider a recursive algorithm defined as follows :*

$$\theta(t+1) = \theta(t) - \mu_t M(\theta(t), \omega(t)). \tag{11}$$

In which, the expectation of $M(\theta, \omega)$ over ω, i.e.

$$\bar{M}(\theta) = \int M(\theta, \omega) P(\omega) d\omega, \tag{12}$$

has unique solution θ^* such that $\bar{M}(\theta^*) = 0$.

Suppose there exists positive constants κ_1 and κ_2 such that the following conditions are satisfied :

(C1) $\mu_t \geq 0$, $\sum_t \mu_t = \infty$ *and* $\sum_t \mu_t^2 < \infty$.
(C2) $\inf_{\varepsilon < \|\theta - \theta^*\| < \varepsilon^{-1}} (\theta - \theta^*)^T \bar{M}(\theta) > 0$, *for all* $\varepsilon > 0$.
(C3) $\int \|M(\theta, \omega)\|^2 P(\omega) d\omega \leq \kappa_1 + \kappa_2 \|\theta\|^2$.

Then for $t \to \infty$, $\theta(t)$ converges to θ^* with probability one.

Normally, the first condition can easily be satisfied. It is because the step size μ_t could be defined as $\frac{\text{const}}{t}$ for all $t \geq 1$. Therefore, we skip the proof of Condition (C1) in the rest of this section. For the sake of presentation, we let $Y = \frac{1}{N} \sum_{k=1}^{N} y_k \phi(x_k)$. Besides, we have $\bar{M}(\theta) = -h(\theta)$ and ω is a random vector augmenting (x_t, y_t, β).

3.1 Multinode Fault Injection Training

Applying Galdyshev Theorem, the following theorem can be proved for injecting multinode fault training.

Theorem 2. *For injecting multinode fault during training an RBF network, the weight vector $\theta(t)$ will converge with probability one to*

$$\theta^* = [H_\phi + p(Q_g - H_\phi)]^{-1} Y. \tag{13}$$

Besides, the corresponding objective function to be minimized is given by

$$\mathcal{L}(\theta|\mathcal{D}) = \frac{1}{N} \sum_{k=1}^{N} (y_k - f(x_k, \theta))^2 + p\theta^T (Q_g - H_\phi)\theta. \tag{14}$$

[1] In the following convergence proof, $\omega(t) = (x_t, y_t, \beta_t)$. Owing not to confuse the time index t with the element index k, the subscript t is omitted. So that $\omega(t) = (x_t, y_t, \beta)$.

3.2 Weight Decay-Based Multinode Fault Injection Training

For weight decay-based multinode fault injection training, we can have the following theorem.

Theorem 3. *For injecting multinode fault during weight decay training an RBF network, the weight vector $\theta(t)$ will converge with probability one to*

$$\theta^* = \left[H_\phi + p(Q_g - H_\phi) + \frac{\lambda}{1-p} I_{M\times M} \right]^{-1} Y. \tag{15}$$

Besides, the corresponding objective function to be minimized is given by

$$\mathcal{L}(\theta|\mathcal{D}) = \frac{1}{N}\sum_{k=1}^{N}(y_k - f(x_k,\theta))^2 + \theta^T \left\{ p(Q_g - H_\phi) + \frac{\lambda}{1-p} I_{M\times M} \right\} \theta. \tag{16}$$

4 Proof of Theorems

Next, we are going to apply the Gladyshev Theorem for the convergence proof. Normally, the first condition can easily be satisfied. It is because the step size μ_t could be pre-defined. So, we skip the proof of Condition (C1) for simplicity and then prove only the Condition (C2) and (C3).

4.1 Injecting Multinode Fault (Theorem 2)

To prove the condition (C2), we need to consider the mean update equation $h(\theta(t))$. By taking the expectation of the second part of the Equation (6) with respect to β_i, x_t and y_t, $h(\theta(t))$ will be given by

$$h(\theta(t)) = \left\{ \frac{1}{N}\sum_{k=1}^{N}(y_k - \phi^T(x_k)\theta(t))\phi(x_k) - p(H_\phi - Q_g)\theta(t) \right\}. \tag{17}$$

In which, the solution θ^* is given by

$$\theta^* = [H_\phi + p(Q_g - H_\phi)]^{-1} Y. \tag{18}$$

Hence, for all $\|\theta - \theta^*\| > 0$, we have

$$-(\theta-\theta^*)^T h(\theta) = -(\theta-\theta^*)^T \left(Y - [H_\phi + p(Q_g - H_\phi)]\,\theta \right),$$

which is greater than zero. Therefore, Condition $(C2)$ is satisfied.

For Condition $(C3)$, we consider the Equation (6). By triangle inequality, it is clearly that

$$\|M(\theta,\omega)\|^2 \leq \|y_t^2 \tilde{\phi}^T(x_t)\tilde{\phi}(x_t)\| + \theta^T(\tilde{\phi}(x_t)\tilde{\phi}^T(x_t))^2\theta. \tag{19}$$

Since,

$$\begin{aligned}\int (\tilde{\phi}^T\tilde{\phi})\tilde{\phi}\tilde{\phi}^T P(\beta)d\beta &\leq \phi^T\phi\lambda_{\max}\left\{\int \tilde{\phi}\tilde{\phi}^T P(\beta)d\beta\right\} \\ &\leq (\phi^T\phi)^2. \end{aligned} \tag{20}$$

Putting Equation (20) into Equation (19), it is clear that

$$\int \left\{\|M(\theta,\omega)\|^2\right\} P(\beta)d\beta \leq \|y_t^2\phi^T(x_t)\phi(x_t)\| + (\phi(x_t)^T\phi(x_t))^2\|\theta\|^2. \tag{21}$$

Further taking the expectation of the above inequality with respect to x_t and y_t, one can readily show that Condition ($C3$) can be satisfied and the proof is completed.

With reference to Equation (17), the constant factor $(1-p)$ can be put together with μ_t and treated as a new step size. Hence, the objective function of the above algorithm is given by

$$\mathcal{L}(\theta|\mathcal{D}) = \frac{1}{N}\sum_{k=1}^{N}(y_k - f(x_k,\theta))^2 + p\theta^T(Q_g - H_\phi)\theta. \tag{22}$$

The proof for Theorem 2 is completed. **Q.E.D.**

It is worthwhile noted that Equation (14) is also identical to the objective function derived for batch model in [16].

4.2 WD-Based Multinode Fault Injection Training (Theorem 3)

The corresponding $h(\theta(t))$ will be given by

$$h(\theta(t)) = (1-p)\left\{\frac{1}{N}\sum_{k=1}^{N}(y_k - \phi^T(x_k)\theta(t))\phi(x_k) - p(H_\phi - Q_g)\theta(t)\right\} - \lambda\theta(t). \tag{23}$$

In which, the solution θ^* is given by

$$\theta^* = \left[H_\phi + p(Q_g - H_\phi) + \frac{\lambda}{1-p}I_{M\times M}\right]^{-1} Y. \tag{24}$$

Hence, for all $\|\theta - \theta^*\| > 0$, we have

$$-(\theta-\theta^*)^T h(\theta) = -(\theta-\theta^*)^T\left(Y - \left[H_\phi + p(Q_g - H_\phi) + \frac{\lambda}{1-p}I_{M\times M}\right]\theta\right),$$

which is greater than zero. Therefore, Condition ($C2$) is satisfied.

For Condition ($C3$), we consider the Equation (9) and the similar technique as for the case of injecting multinode fault, one can readily show that

$$\int \left\{\|M(\theta,\omega)\|^2\right\} P(\beta)d\beta \leq \|y_t^2\phi^T(x_t)\phi(x_t)\| + \left\{(\phi(x_t)^T\phi(x_t))^2 + \lambda^2\right\}\|\theta\|^2. \tag{25}$$

Further taking the expectation of the above inequality with respect to x_t and y_t, one can readily show that Condition $(C3)$ can be satisfied and the proof is completed.

With reference to Equation (23), the objective function is given by

$$\mathcal{L}(\theta|\mathcal{D}) = \frac{1}{N}\sum_{k=1}^{N}(y_k - f(x_k,\theta))^2 + \theta^T \left\{ p(Q_g - H_\phi) + \frac{\lambda}{1-p} I_{M\times M} \right\} \theta. \quad (26)$$

The proof for Theorem 3 is completed. **Q.E.D.**

One should notice that the weight decay effect is scaled up when random node fault is injected.

5 Conclusions

In this paper, proofs on the convergences of two node-fault-injection-based online training RBF methods have been shown and their corresponding objective functions have been deduced. For the injecting multinode-fault training, it is also found that the objective function is identical to the one that is proposed in [16] for batch-mode training an RBF to deal with multinode fault.

For the weight decay-based multinode fault injection training, two additional regularization terms are obtained in the objective function. The first one is identical to the extra term obtained for pure multinode fault injection training. The other is a weight decay term with a constant factor $\lambda/(1-p)$ is depended on the fault rate p. The constant factor can amplify the penalty of weight magnitude if the fault rate is large.

It is worthwhile noted that for λ not equal to zero, regularization effect will still exist in null fault rate situation. Generalization can be improved. So, it is suspected that weight decay-based multimode fault injection training might lead to network model with good generalization and multinode fault tolerance ability. Further investigation along the line should be valuable for future research.

Acknowledgement

The research work reported in this paper is supported in part by Taiwan NSC Research Grant 97-2221-E-005-050-.

References

1. An, G.: The effects of adding noise during backpropagation training on a generalization performance. Neural Computation 8, 643–674 (1996)
2. Bernier, J.L., et al.: Obtaining fault tolerance multilayer perceptrons using an explicit regularization. Neural Processing Letters 12, 107–113 (2000)
3. Bishop, C.M.: Training with noise is equivalent to Tikhonov regularization. Neural Computation 7, 108–116 (1995)

4. Bolt, G.: Fault tolerant in multi-layer Perceptrons. PhD Thesis, University of York, UK (1992)
5. Cavalieri, S., Mirabella, O.: A novel learning algorithm which improves the partial fault tolerance of multilayer NNs. Neural Networks 12, 91–106 (1999)
6. Chandra, P., Singh, Y.: Fault tolerance of feedforward artificial neural networks – A framework of study. In: Proceedings of IJCNN 2003, vol. 1, pp. 489–494 (2003)
7. Chiu, C.T., et al.: Modifying training algorithms for improved fault tolerance. In: ICNN 1994, vol. I, pp. 333–338 (1994)
8. Deodhare, D., Vidyasagar, M., Sathiya Keerthi, S.: Synthesis of fault-tolerant feedforward neural networks using minimax optimization. IEEE Transactions on Neural Networks 9(5), 891–900 (1998)
9. Gladyshev, E.: On stochastic approximation. Theory of Probability and its Applications 10, 275–278 (1965)
10. Grandvalet, Y., Canu, S.: A comment on noise injection into inputs in backpropagation learning. IEEE Transactions on Systems, Man, and Cybernetics 25(4), 678–681 (1995)
11. Grandvalet, Y., Canu, S., Boucheron, S.: Noise injection : Theoretical prospects. Neural Computation 9(5), 1093–1108 (1997)
12. Hammadi, N.C., Hideo, I.: A learning algorithm for fault tolerant feedforward neural networks. IEICE Transactions on Information & Systems E80-D(1) (1997)
13. Ho, K., Leung, C.S., Sum, J.: On weight-noise-injection training. In: Köppen, M., Kasabov, N., Coghill, G. (eds.) ICONIP 2008. LNCS, vol. 5507, pp. 919–926. Springer, Heidelberg (2009)
14. Kamiura, N., et al.: On a weight limit approach for enhancing fault tolerance of feedforward neural networks. IEICE Transactions on Information & Systems E83-D(11) (2000)
15. Lai, T.L.: Stochastic approximation. Annals of Statistics 31(2), 391–406 (2003)
16. Leung, C.S., Sum, J.: A fault tolerant regularizer for RBF networks. IEEE Transactions on Neural Networks 19(3), 493–507 (2008)
17. Neti, C., Schneider, M.H., Young, E.D.: Maximally fault tolerance neural networks. IEEE Transactions on Neural Networks 3(1), 14–23 (1992)
18. Phatak, D.S., Koren, I.: Complete and partial fault tolerance of feedforward neural nets. IEEE Transactions on Neural Networks 6, 446–456 (1995)
19. Reed, R., Marks II, R.J., Oh, S.: Similarities of error regularization, sigmoid gain scaling, target smoothing, and training with jitter. IEEE Transactions on Neural Networks 6(3), 529–538 (1995)
20. Sequin, C.H., Clay, R.D.: Fault tolerance in feedforward artificial neural networks. Neural Networks 4, 111–141 (1991)
21. Sum, J., Leung, C.S., Ho, K.: On objective function, regularizer and prediction error of a learning algorithm for dealing with multiplicative weight noise. IEEE Transactions on Neural Networks (accepted for publication)
22. Takase, H., Kita, H., Hayashi, T.: A study on the simple penalty term to the error function from the viewpoint of fault tolearnt training. In: Proc. IJCNN 2004, pp. 1045–1050 (2004)
23. Tchernev, E.B., Mulvaney, R.G., Phatak, D.S.: Investigating the Fault Tolerance of Neural Networks. Neural Computation 17, 1646–1664 (2005)

Part III

Kernel Methods and SVM

Symbolic Knowledge Extraction from Support Vector Machines: A Geometric Approach

Lu Ren and Artur d' Avila Garcez

City University, London, UK
{cf529,aag}@soi.city.ac.uk

Abstract. This paper presents a new approach to *rule extraction* from Support Vector Machines (SVMs). SVMs have been applied successfully in many areas with excellent generalization results; rule extraction can offer *explanation capability* to SVMs. We propose to approximate the SVM classification boundary by solving an optimization problem through sampling and querying followed by boundary searching, rule extraction and post-processing. A theorem and experimental results then indicate that the rules can be used to validate the SVM with high accuracy and very high fidelity.

1 Introduction

In recent years, artificial neural networks (ANNs) [2,3,8] have been utilised in many applications offering excellent prediction results. In many cases, however, developers prefer not to use ANNs because of their inability to explain how the results have been obtained. *Rule extraction* addresses the above problem. By extracting rules from ANNs, we can explain the reasoning process and validate the learning system. Rule extraction helps, therefore, to integrate the symbolic and connectionist approaches to AI, offering ways of combining the statistical nature of learning with the logical nature of reasoning.

Recently, SVMs started to be considered for rule extraction because of their excellent generalization capability and classification accuracy. Angulo et al [10] used support vectors and prototypes to draw regions indicating equation rules or interval rules. Barakat et al [9] used support vectors to construct synthetic data, feed the data into a decision tree learner, and extract rules. Fung et al [4] extracted a set of rules that approximate linear SVM hyperplanes.

In our opinion, a satisfactory extraction method, striking a balance between the need for correctness and generalization, is still needed. In this paper, we tackle this issue by proposing a new any-time rule extraction algorithm, which uses the SVM as an oracle (black-box) and synthetic data for querying and rule extraction, thus making very few assumptions about the training process and the SVM training data. Hence, the algorithm does not depend on the availability of specific training sets; it captures the information encoded in the geometry of the SVM classification boundary through querying, clustering, and searching. It then performs rule extraction, solving an optimization problem, and rule post-processing. Also, our approach is not restricted to any specific SVM classifier such as the linear classifier considered in [4]. It is in fact applicable to most neural-network models. We have examined rule accuracy, fidelity and comprehensibility in three applications: the Monk's problems, the iris flower dataset and the breast

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 335–343, 2009.

cancer-wisconsin dataset. The results indicate the soundness of our approach through maximum fidelity.

2 Support Vector Machines

We consider the problem of classifying *n* points in the *m*-dimensional input space R^m. Consider the training data set $\{(\mathbf{x}_i, y_i)\}$, $i = 1, ..., n$, $y_i \in \{1, -1\}$ and $\mathbf{x}_i \in R^m$. In the case of linear SVMs the decision function is a separating hyperplane $f(x) = sign(w \cdot x + b)$. The optimal classification hyperplane that maximizes the distance between class $A_+ = 1$ and $A_- = -1$ can be found by *minimizing* $1/2\|w\|^2$ *subject to* $y_i(w \cdot \mathbf{x}_i + b) \geq 1$.

The Lagrangian J below has been introduced to solve this problem: $J = \frac{1}{2}w^T w - \Sigma_{i=1}^{n} \alpha_i (y_i(w \cdot \mathbf{x}_i + b) - 1)$, where $\alpha_i \geq 0$ is known as the lagrange multiplier. With respect to w and b, minimizing J leads to $w = \Sigma_{i=1}^{sv} \alpha_i y_i \mathbf{x}_i$ and $\Sigma_{i=1}^{n} \alpha_i y_i = 0$ where sv is the number of support vectors [12]. By making some substitutions, the hyperplane decision function ($f(x) = sign(\Sigma_{i=1}^{sv} \alpha_i y_i \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle + b)$, $j = 1, ..., n$) can be obtained, where $\langle\rangle$ denotes inner product. For nonlinear classification, the decision function can be $f(x) = sign(\Sigma_{i=1}^{sv} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b)$. See [12] for more on SVMs.

3 Geometric SVM Rule Extraction

Most rule extraction algorithms suffer from a lack of generality. In this section, we present a novel algorithm called *Geometric Oracle-Based SVM Rule Extraction* (GOSE), which is designed to alleviate this limitation. GOSE utilizes the points on the SVM classification boundary and synthetic training instances to construct a set of optimized hypercube rules. The area covered by those rules is maximized, and it can be proved that the rules approximate the SVM.

GOSE aims to use the classification boundary and synthetic training instances to extract the hypercube rules without considering the inner structure and the support vectors. It treats the SVMs as *oracles*, [1] making few assumptions about the architecture and training process, hopefully being applicable to other non-symbolic learning methods. All that is assumed is that an SVM is given which we can query and find the classification y_i that it produces for input vectors $\mathbf{x}_i$. After querying, by means of a kind of binary search, we look for the points $\mathbf{P}$ that lie on the SVM classification boundaries. Subsequently, an initial optimal rule set can be extracted for the points in $\mathbf{P}$ and the synthetic training instances by solving an optimization problem whereby we attempt to find the largest consistent hypercubes in the input space. Finally, several post-processing measures are applied to the rule set in order to derive (a relatively small number of) generalized rules. In what follows, we explain each of the above steps.

Querying. GOSE is a general method because it uses a subset of synthetic training inputs to query the SVM and obtain the class labels for the inputs. We use a random data

[1] The querying process makes our approach independent of any special training data, no assumption is made about the structure of the network, and the approach can be applied to any SVM classifier, regardless of the algorithms used to construct the classifier.

generator to produce a large amount of inputs. Unlike TREPAN [8], GOSE uses *multivariate kernel density estimates* which take into account the *relations within features*. The probability density function for inputs $\mathbf{x}$ is:

$$M(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} \frac{1}{\prod_{j=1}^{m} h_j}\left[\frac{1}{(\sqrt{2\pi})^m} e^{-\frac{1}{2}(\|\frac{\mathbf{x}-\mathbf{X}_i}{h}\|)}\right]$$

where $\mathbf{X}_i$ are the training samples, $1 \leq i \leq N$. $h = [h_1, h_2, ...h_m]$ is a vector of bandwidth such that $h = [\frac{4}{(m+2)N}]^{\frac{1}{m+4}}\sigma$, where σ is the standard deviation of the training samples. After obtaining those synthetic inputs $\{\mathbf{x}_i, i \geq 1\}$, we treat them as the inputs of the SVM, i.e. the SVM is used as an *oracle*.

One key issue in querying is how to know how many classes there are. GOSE generates a large amount of data j uniformly distributed in the input space. This dataset is then given as input to the SVM to acquire class labels. For a large j, the dataset should spread throughout the input space evenly. Then, a set of classes $A = \{A_k | 1 \leq k \leq CN\}$, where CN is the total number of classes, can be obtained.

Clustering. Since there must be a classification boundary between different classes, we can find the points lying on the boundary between pairs of data for different classes. However, for a large number of training data, if we search each pair, this may lead to computational complexity issues. Hence, we use clustering and search for the boundary between pairs of data clusters instead. Our goal is to strike a balance between algorithmic complexity and predictive accuracy. A cluster C can be defined as a subset of training data $S = \{(\mathbf{x}_i, y_i)\}$, with the same class y_i.

We use hierarchical clustering on S. Our approach uses one of the following linkage functions: *single linkage*, uses the smallest distance between data $\mathbf{x}_i^r$ and $\mathbf{x}_j^s$ in the two clusters r and s; *complete linkage*, uses the largest distance between data $\mathbf{x}_i^r$ and $\mathbf{x}_j^s$. Given a set of clusters $\{r_h,\ h = 1, 2, ..., q\}$, where the classes of the clusters are identical, let the number of data points in each cluster r_h be n^{r_h}. It is obvious that the mean and variance of each cluster relate to the data $\mathbf{x}_i, 1 \leq i \leq n^{r_h}$. The mean m^{r_h} of each cluster r_h is $m^{r_h} = \frac{1}{n^{r_h}}\sum_{i=1}^{n^{r_h}} \mathbf{x}_i$, and the variance $s^{r_h} = \frac{1}{n^r}\sum_{i=1}^{n^{r_h}}(\mathbf{x}_i - m^{r_h})^2$.

Hence, the intra-cluster deviation is:

$$s^{intra} = \sqrt{\sum_{h=1}^{q}(s^{r_h} * p(r_h))} \;\; where \;\; p(r_h) = \frac{n^{r_h}}{\sum_{h=1}^{q} n^{r_h}} \qquad (1)$$

and the inter-cluster mean and deviation are:

$$m^{inter} = \sum_{h=1}^{q}(m^{r_h} * p(r_h)) \qquad s^{inter} = \sqrt{\sum_{h=1}^{q}[(m^{r_h} - m^{inter})^2 * p(r_h)]} \qquad (2)$$

Our stopping criterion is the rate between s^{intra} and s^{inter}. If $\frac{s^{intra}}{s^{inter}} > \epsilon$, then GOSE will stop clustering the data further. Note that ϵ is a user-defined parameter.

Searching. The searching step locates the points on the SVM decision boundaries. Given clusters $P_1, P_2, ...P_a$ which fall into class A_+, and clusters $N_1, N_2, ...N_b$ which fall into class A_-, we use Zhang and Liu's measure [14] to automatically look for the points on the SVM's decision boundary[2].

We consider all pairs (p, n) s.t. $p \in P_j(1 \leq j \leq a)$ and $n \in N_k(1 \leq k \leq b)$. For each p, we find a corresponding point n whose distance to p is minimum. And for each n, we find a corresponding point p whose distance to n is minimum. Let d_1 represent the distance from p to the hyperplane and d_2 represent the same for n. In order to find the point lying on the hyperplane, a kind of binary search is performed on (p, n). In other words, if $|d_1 - d_2| > \varepsilon$, the mid-point q between p and n is chosen. The SVM classifies q and computes the distance between q and the hyperplane. If the class of q equals that of p, then p is replaced by q; otherwise, n is replaced by q. The process carries on until $|d_1 - d_2| < \varepsilon$ is achieved, where ε denotes an arbitrary small number.

Extraction. The main idea of our approach is to find a set of optimal rules that (1) covers a maximum area, and (2) covers the largest cardinality of synthetic instances. Suppose that a set of points X lie on the SVM decision boundary, where X is the result of *searching*, a set of synthetic training instances S for classes $A = \{A_p, 1 \leq p \leq CN\}$ has been generated from *querying*, and let $f(\mathbf{x})$ be the SVM function. To realize the first goal of the rule extraction algorithm, we try to solve the following optimization problem:

$$maximize \qquad \prod_{i=1}^{m}(x_i - \mathbf{x}_{0_i}) \tag{3a}$$

$$subject\ to \qquad l \leq \mathbf{x} \leq u \tag{3b}$$

$$\int_l^u (f(\mathbf{x}) - A_p)d\mathbf{x} = 0 \tag{3c}$$

where $\mathbf{x}_{0_i}$ denotes the i^{th} element of vector $\mathbf{x}_0 \in X$ ($\mathbf{x}_0$ indicates a starting point), x_i is the i^{th} element of $\mathbf{x}$, l and u are the m-dimensional vectors giving lower and upper bounds to this optimization problem.

The objective function (Equation 3a) aims to maximize the volume of the hypercube that a rule covers, and it has two constraints. One is a bound constraint to limit the optimal $\mathbf{x}^*$ in a given area, while the other is a nonlinear constraint that is used to exclude the points that have different class labels.

The values of l and u in Equation 3b can be calculated based on the lower and upper bounds of the input space. For example, suppose the scope of the input space is $[L_1, L_2] \leq \mathbf{x} \leq [U_1, U_2]$, and $\mathbf{x}_0 = [\mathbf{x}_{0_1}, \mathbf{x}_{0_2}]$ is a point lying on the SVM boundary. When we change $\triangle\mathbf{x}_{0_1}$ on $\mathbf{x}_{0_1}$ or $- \triangle \mathbf{x}_{0_2}$ on $\mathbf{x}_{0_2}$ ($\triangle\mathbf{x}_{0_1}, \triangle\mathbf{x}_{0_2} \geq 0$) the SVM classification is still class A_p. Hence, it is reasonable to assume that an optimal point can be found in the rectangle between points $\mathbf{x}_0$ and $[U_1, L_2]$. l and u are then narrowed down to $l = [\mathbf{x}_{0_1}, L_2]$ and $u = [U_1, \mathbf{x}_{0_2}]$.

As presented in Equation 3c, the nonlinear constraint is a multi-dimension integral on a linear/nonlinear function. GOSE uses quasi-Monte Carlo [13] to approximate the integration because it is a method with advantages such as improved convergence and more uniformity. Therefore, the hypercube H is considered to be composed of the points that

[2] Notice that for simplicity we have been considering two classes, but our extraction algorithm is applicable to any number of classes.

are uniformly distributed as: $\frac{1}{n}\sum_{i=1}^{n}|f(\mathbf{a}_i)-A_p| \approx \int_l^u |f(\mathbf{x})-A_p|\, d\mathbf{x}$, where $\mathbf{a}_i$ is a low-discrepancy sequence inside the hypercube $[l,u]$, where $1 \leq i \leq n$, and n here means the number of points selected for approximation in H. The estimation error then becomes $\epsilon = \left|\int_l^u |f(\mathbf{x})-A_p|\, dx - \frac{1}{n}\sum_{i=1}^{n}|f(\mathbf{a}_i)-A_p|\right|$.

From the above, it can be shown that the larger n is, the closer the approximation gets to the integral. It is clear that the complexity increases with n. Therefore, in order to strike a balance between error estimation, fidelity, accuracy prediction and complexity, a proper n has to be chosen. In the cross-validation experiments below, we found that $n = 1000$ was a suitable number for the benchmark datasets.

Finally, the standard *pattern search algorithm* is applied to obtain a solution $\mathbf{x}^*$ to the optimization problem (see [1] for an analysis of generalized pattern search by evaluating the objective function). After obtaining $\mathbf{x}^*$, together with the starting point $\mathbf{x}_0$, the antecedents of a rule can be constructed by picking the minimum and maximum values of $\mathbf{x}^*$ and $\mathbf{x}_0$. The set of rules covering all of the synthetic training instances can be found by repeatedly applying Equation 3, but with $\mathbf{x}_0$ replaced by $s \in S$. The process ensures that the extracted rules cover most of the training instances and the maximum area for each rule.

Post-Processing. The rule set extracted above may still contain overlapping rules. The purpose of post-processing is to prune rules with high error estimation and to construct non-overlapping rules with high coverage/generalization rate.

Given two rules, $r_1 = \bigwedge(a_i \leq x_i \leq b_i) \rightarrow A_p$ and $r_2 = \bigwedge(c_i \leq x_i \leq d_i) \rightarrow A_p$, r_1 and r_2 are said to be non-overlapping if $b_i \leq c_i$ or $a_i \geq d_i$ for any i, $1 \leq i \leq m$.

1) Rule Extending. Let the input space of a problem domain range from L_i to U_i, and consider rule $r = \bigwedge(l_i \leq x_i \leq u_i) \rightarrow A_p$. The rule-extending step attempts to extend the interval of r on each dimension into a larger scope. At the same time, the new rule r' still has to satisfy the constraint that the area covered by r' belongs to class A_p.

The complexity of this step is exponential on the size of the input space. Hence, in practice, another optimizing measure has been adopted. If the new region of a rule consists of the points for another class when the rule is extended on a certain dimension j then this dimension will be skipped in the rest of the iteration.

Finally, if the value of the i^{th} dimension is extended up to the scope of the whole input space, which is believed to be applicable throughout the range of the i^{th} dimension, GOSE then removes this dimension from the antecedent of the rule.

2) Rule Pruning. This step aims to prune the rules that have a relatively large estimated error. GOSE uses a *t-test* to analyze the null hypothesis that the mean of the estimated value and the expected value of the integral of a rule r are equal to zero. Morohosi and Fushimi [5] introduced a statistical method for quasi-Monte Carlo error estimation. The rule pruning of GOSE is based on this method.

After the estimate of the mean $\hat{I}$ and the variance $\hat{\sigma^2}$ are obtained,the *t-test* turns out to be: $t = \frac{\hat{I}}{\frac{\hat{\sigma}}{\sqrt{M}}}$ If t is larger than the standard value at the significance level set by GOSE, the rule is rejected. Otherwise it is accepted. Those rejected rules are removed from the rule set.

3) Non-overlapping Rule Construction. As mentioned above, there could exist overlapping rules. This step removes the intersections between rules to improve the comprehensibility of rules. We want at least one dimension of each of two rules not to intersect with each other. Let r_1 be $[a_1, .., a_m] \leq \mathbf{x} \leq [b_1, ..., b_m] \rightarrow A_p$ and r_2 be $[c_1, ...c_m] \leq \mathbf{x} \leq [d_1, .., d_m] \rightarrow A_p$. If $a_i \leq c_i \leq b_i \leq d_i$, $1 \leq i \leq m$, then the overlap of r_1 and r_2 is $\{[c_1, .., c_m] \leq \mathbf{x} \leq [b_1, ..b_m]\}$. Suppose r_2 does not change and r_1 has to be divided. For each dimension i, a non-overlapping rule can be constructed in three steps:

1. Keep the original value $a_j \leq x_j \leq b_j$ of r_1, $j < i$.

2. Take the non-overlapping value $a_i \leq x_i \leq c_i$, $j = i$.

3. Replace the original values with the overlapping values $c_j \leq x_j \leq b_j$, $j > i$.

4) Rule Selection. The last step of post-processing is *rule selection*. This discards those rules with zero coverage rate. This means that the rules predicting no data in our experiments are removed.

Although the extraction algorithm is based on querying the SVM, the approximations associated with Monte Carlo and post-processing may render it incomplete. We need to make sure, however, that - as the rule extraction algorithm is applied - the set of rules converges to the actual behaviour of the SVM, i.e. the rules approximate the SVM. Since this is a geometric approach, it suffices to show that the area covered by the set of rules approaches the area as classified by the SVM. We call this *quasi-completeness*.

Theorem (quasi-completeness): *With an increasing number of rules, the rule set approximates the SVM. Let S denote the area covered by the non-overlapping rule set $R = \{r_i \rightarrow A_p, i \geq 1\}$ and V represent the area of interest for the given SVM. When the number of rules increases, S approximates V, that is $\frac{|V-S|}{V} \leq \epsilon$, where ϵ is an arbitrary small number. Note that r_i refers to an area where class A_p occurs.*[3]

4 Experimental Results

We performed experiments using three datasets, all obtained from the UCI Machine Learning repository: the Iris flower dataset, the Breast Cancer-Winsconsin dataset, and the Monk's problems. 5-fold cross validation has been used. For each fold:

1. We trained the SVM using different algorithms: DAGs-SVM [6] for the Iris dataset and Monk-2, and SMO [7] for Breast Cancer, Monk-1 and Monk-3 problems.
2. We generated a number of training data and queried the trained SVM to obtain the class labels.
3. We applied GOSE to datasets of varying sizes using varying numbers of clusters.
4. We measured rule accuracy with respect to the test set, rule fidelity to the SVM, and rule comprehensibility w.r.t. the number of rules extracted and the number of antecendents per rule.

Accuracy measures the ability of the rules in predicting unseen cases according to a test set. The results show that when the data size[4] N increases, the accuracy of the rules increases, converging to that of the SVM as illustrated in Figure 1 (left). For example, for the Iris dataset, when N equals 30, the accuracy is only 77.33%. However, 84.67%

[3] Proofs of soundness and quasi-completeness can be found in a technical report which is available at `http://www.soi.city.ac.uk/~aag/papers/gose.pdf`

[4] The data set here indicates the synthetic set so that the data size is not limited to that of the original training set.

is achieved when N equals 100. It finally reaches 89.33% at $N = 300$, which is a value near to the accuracy of the SVM. The same behavior is verified for the breast cancer dataset and Monk's problems. For the breast cancer dataset, when N equals 50, the accuracy is only around 60%. Although the rate of the increase reduces, the accuracy finally reaches 90.14% at $N = 200$. For the Monk's problem, when N reaches 100, GOSE achieves 100% accuracy for Monk-1. For Monk-2, GOSE obtains 84.8% for a 200-size training set compared with the 85.7% accuracy classified by the SVM network. GOSE also achieves an average 95% correctness for a 100-size training set in the case of the Monk-3 problem, while the SVM obtains around 94% accuracy.

Figure 1 (right) shows that when the number of clusters increases, the accuracy increases as well. As an example, in the Iris dataset, GOSE classifies only 56% instances correctly when the cluster number is one. But it predicts 84% instances correctly when the number of clusters goes to six. It is interesting to note that the value of 84%, closest to the SVM accuracy of 89.33%, is obtained when the training set contains 300 instances and has one cluster for each class. The same convergence movement occurs for the Breast Cancer dataset. An accuracy of only 62.23% is obtained when each class has one cluster, but 87.55% of instances are predicted correctly when the number of clusters goes to six. For the Monk's problem, the accuracy is only 97%, 62% and 58% respectively for Monk-1, Monk-2 and Monk-3, when the number of clusters is one. However, the accuracy increases to 100% for Monk-1, 78% for Monk-2 and 90% for Monk-3, when the number of clusters increases to 5. This is why we have chosen to use the stopping criterion of Section 3 to find an appropriate cluster value.

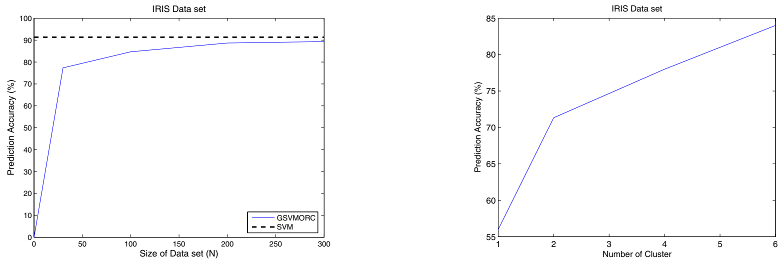


Fig. 1. Results of Iris by changing the dataset size & cluster number

Fidelity measures how close the rules are to the actual behavior of the SVM, as opposed to its accuracy w.r.t a test set. The fidelity in Monk-1, Iris and Breast Cancer problems has been 100%. Monk-2 and Monk-3 obtained 99.12% and 98.5% fidelity.

Comprehensibility measures the number of rules and the number of conditions per rule. The following is an example of the extracted rules on the Iris dataset. GOSE obtains on average ten rules for each class, with four conditions per rule.

$$\textit{sepal length} = [4.3, 6.6] \bigwedge \textit{sepal width} = [2.0, 4.0] \bigwedge \textit{petal length} = [2.7, 5.0] \bigwedge \textit{petal width} = [0.4, 1.7] \rightarrow \textit{Iris Versicolour}$$

The rule above correctly predicts 45 out of 150 instances in the data set. Overall, 93% training and test examples in the Iris data set are predicted correctly.

The following is an example of the extracted rules on Breast Cancer problem. GOSE obtained on average 26 rules for class 1 and 81 rules for class -1, with an average 7.2 conditions per rule. The final rule set classifies 90.14% of the test cases and 93% of the whole data set correctly.

$$a_3 = [4,9] \bigwedge a_5 = [3,9] \bigwedge a_6 = [10,10] \bigwedge a_7 = [5,9] \rightarrow -1$$

For Monk-1, GOSE obtained four rules for all classes. On average, each rule has 2.37 conditions. This is from the 100 synthetic training instances, which cover 100% of the test cases. For Monk-2, 38 rules were extracted, with around 4.1 conditions per rule for class 1. For class 0, 24 rules with 5.8 conditions per rule were extracted. For Monk-3, 11 rules were extracted, with around 3.4 conditions per rule for class 1, while 6 rules with 2.7 conditions per rule were extracted for class 0 (for example $a_1 = 1 \bigwedge a_2 = 1 \rightarrow class = 1$; in this case, $a_1 = 1$ denotes that a_1 is $true$).

5 Conclusion and Future Work

We have presented an effective algorithm for extracting rules from SVMs so that results can be interpreted by humans more easily. A key feature of the extraction algorithm is the idea of trying to search for *optimal rules* with the use of expanding hypercubes, which characterize rules as constraints on a given classification. The main advantages of our approach are that we use synthetic training examples to extract accurate rules, and treat the SVM as an oracle so that the extraction does not depend on specific training requirements or given training data sets. Empirical results on real-world data sets indicate that the extraction method is correct, as it seems to converge to the true accuracy of the SVM as the number of training data increases and high fidelity rates are obtained in every experiment. In future work, we may consider using different shapes than hypercubes for the extraction of rules and compare results and further improve the comprehensibility of the extracted rules.

Support Vector Machines have been shown to provide excellent generalization results and better classification results than parametric methods or neural networks as a learning system in many application domains. In order to develop further the study of the area, we need to understand why this is so. Rule extraction offers a way of doing this by integrating the statistical nature of learning with the logical nature of symbolic artificial intelligence [11].

References

1. Charles, A., Dennis, J.E.: Analysis of generalized pattern searches. SIAM Journal on Optimization 13(3), 889–903 (2003)
2. Garcez, A.A., Broda, K., Gabbay, D.: Symbolic knowledge extraction from trained neural networks: a sound approach. Artificial Intelligence 125(1-2), 155–207 (2001)
3. Thrun, S.B.: Extracting provably correct rules from artificial neural networks. Technical report, Universität Bonn (1993)
4. Fung, G., Sandilya, S., Rao, B.R.: Rule generation from linear support vector machines. In: KDD 2005, pp. 32–40 (2005)

5. Morohosi, H., Fushimi, M.: A practical approach to the error estimation of quasi-monte carlo integration. In: Niederreiter, H., Spanier, J. (eds.) Monte Carol and Quasi-Monte Carlo Methods, pp. 377–390. Springer, Berlin (1998)
6. Platt, J., Cristianini, N., Shawe-Taylor, J.: Large margin dags for multiclass classification. Advances in Neural Information Processing Systems 12, 547–553 (2000)
7. Platt, C.J.: Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research (April 1998)
8. Craven, W.M., Shavlik, J.W.: Using sampling and queries to extract rules from trained neural networks. In: International Conference on Machine Learning, pp. 37–45 (1994)
9. Barakat, N., Diederich, J.: Eclectic rule extraction from support vector machines. International Journal of Computational Intelligence 2(1), 59–62 (2005)
10. Nũñez, N., Angulo, C., Catalá, A.: Rule extraction from support vector machines. In: Proceeding of European Symposium on Artificial Neural Networks, Bruges, Belgium, pp. 107–112 (2003)
11. Leslie, G.V.: Three problems in computer science. J. ACM 50(1), 96–99 (2003)
12. Vapnik, N.V.: Statistical learning theory. John Wiley and Sons, INC, Chichester (1998)
13. Morokoff, J.W., Caflisch, R.E.: Quasi-Monte Carlo integration. J. Comp. Phys. 122, 218–230 (1995)
14. Zhang, J.Y., Liu, Y.X.: SVM decision boundary based discriminative subspace induction. Pattern Recognition 38(10), 1746–1758 (2005)

Asbestos Detection from Microscope Images Using Support Vector Random Field of Local Color Features

Yoshitaka Moriguchi, Kazuhiro Hotta, and Haruhisa Takahashi

The University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo 182-8585, Japan
{moriguchi,hotta,takahasi}@ice.uec.ac.jp

Abstract. In this paper, an asbestos detection method from microscope images is proposed. The asbestos particles have different colors in two specific angles of the polarizing plate. Therefore, human examiners use the color information to detect asbestos. To detect the asbestos by computer, we develop the detector based on Support Vector Machine (SVM) of local color features. However, when it is applied to each pixel independently, there are many false positives and negatives because it does not use the relation with neighboring pixels. To take into consideration of the relation with neighboring pixels, Conditional Random Field (CRF) with SVM outputs is used. We confirm that the accuracy of asbestos detection is improved by using the relation with neighboring pixels.

1 Introduction

It is reported that many employees of the company which was manufacturing the materials using asbestos had passed away, and the influence on a human body with asbestos came to be recognized widely. In Japan, the manufacture of the asbestos content products was forbidden, and the amount of the asbestos has been reduced. However, asbestos were used as building materials. Therefore, we need to check whether asbestos are used or not when the buildings are demolished. However, there are still few inspection organizations of asbestos in building materials. Moreover, human examiners must check whether asbestos are included or not, and the number of inspection per a day is also restricted. We must develop an automatic asbestos detector using computers.

There are some automatic asbestos detection methods [10,11,12]. However, they detect only at the atmospheric airborne asbestos. We pay attention to the asbestos detection problem in building materials as opposed to atmosphere. The asbestos detection in building materials is more difficult than that in atmosphere because the specimen from the building materials includes various kinds of particles. In near future, the asbestos detection problem in building materials will become important more and more because the materials including asbestos have been used in school and public buildings in Japan. This paper proposes an automatic asbestos detection method based on statistical pattern recognition.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 344–352, 2009.

The asbestos qualitative analysis by human examiners is called as the disperse dyeing method. Disperse dyeing method prepares three specimens from one sample, and 1000 particles are counted from by each specimen. When more than four asbestos are contained in 3000 particles, it judged as dangerous. The asbestos particles have different color in two specific angles of the polarizing plate. Human examiners detect asbestos using the unique dispersion property of asbestos. This property is also effective to detect asbestos by computer.

In the experiments, we use the two phase-contrast microscope images which are set in two angles of the polarizing plate. By capturing images with two different angles, the position gap between two images arises. At first, we must correct the position gap. We can use the distance between two images. However, since the asbestos particles have different color in two images, we must eliminate the influence of asbestos in the evaluation measure. To do so, the distance between two images is computed on several local regions instead of whole image. In order to determine the characteristic local regions to compute the distance, Scale Invariant Feature Transform (SIFT) [1] is used. The distance between two images is calculated by using the selected local regions, and the position gap is corrected so that the distance is minimized.

After correcting the gap, the asbestos particles are detected. Since the asbestos particles have specific colors in two phase-contrast microscope images, local color features are effective. We develop the detector using SVM [4,7] of local color features. However, when the detector is applied to each pixel independently, there are many false positives and negatives. This is because the relation with neighboring pixels is not taken into consideration. In order to improve this, CRF [6] with SVM outputs is used. By using CRF based on SVM outputs, the accuracy of asbestos detection is improved.

In section 2, the proposed asbestos detection method is explained. Experimental results are shown in section 3. Section 4 describes conclusion and future works.

2 Proposed Asbestos Detection Method

This section explains how to develop a part of disperse dyeing method by computer. At first, the outline of the disperse dyeing method is described in section 2.1. Section 2.2 explains the image matching method for correcting the position gap. Next, the proposed asbestos detection method based on CRF based on SVM outputs is explained in section 2.3. Section 2.4 explains details of the proposed method.

2.1 Disperse Dyeing Method

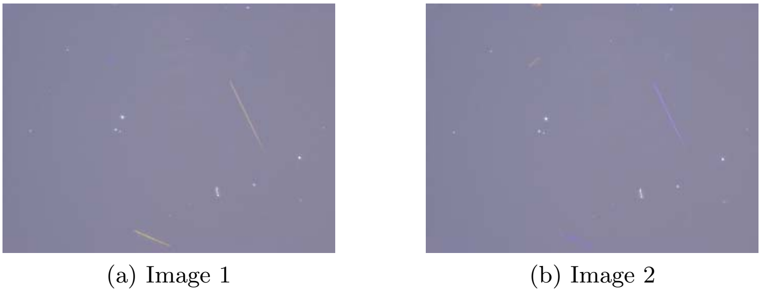
In the disperse dyeing method, a human examiner observes the phase-contrast microscope attached a polarizing plate to each of three specimens. The examiner counts the particles. When a polarizing plate is adjusted, the particles whose the color are changed are recognized as asbestos. In Japanese Industrial Standards, when more than four asbestos are included in 3,000 particles, it is judged as dangerous. This paper proposes an automatic asbestos detection method from two phase-contrast microscope images with difference angles of the polarizing plate.

2.2 Image Matching for Correcting the Position Gap

In this paper, asbestos detection is carried out using the following two images.

Image 1 and 2. The image of a phase-contrast microscope which sets the polarizing plate as the angle θ_1 and θ_2

Figure 1 shows examples of image 1 and 2. The long and thin particles whose color changes in two images are asbestos. The position gap of particles in two images arises by changing the angle of a polarizing plate. Thus, we must correct the gap in two images.


(a) Image 1

(b) Image 2

Fig. 1. Example of image 1 and 2

Figure 1 shows that the color of asbestos changes in two microscope images. Therefore, if the distance of the whole image is used as the evaluation measure to correct the gap, the gap is not corrected well. This is because the distance on asbestos becomes large at the correct position. The distance is calculated using some characteristic local regions instead of whole image. To do so, feature points are detected from the image 1, and the distance is calculated using the detected local regions. SIFT is used for detecting of feature points. Example of the detected feature points from Figure 1 is shown in Figure 2. Red points represent the detected feature points. Many points are selected on large particles. To compute the distance using the various particles fairly, feature points are selected at a certain interval so that the number of few points is 10. The distance of two images is computed by using the color information (RGB) of the local region with 21×21 pixels whose center is the selected feature points. We evaluated the distance while changing the position of image 2. When the distance is minimized, we judge that the position gap is corrected. Then the difference image between image 1 and 2 is computed at the corrected position, and the difference image is binarized by a threshold (In experiments, threshold is set to 0.02.). Example

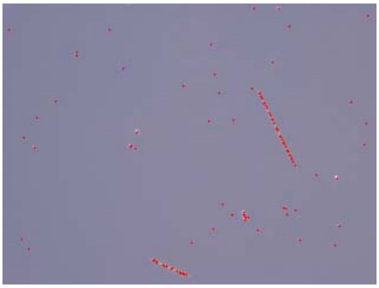


Fig. 2. Example of detected feature points

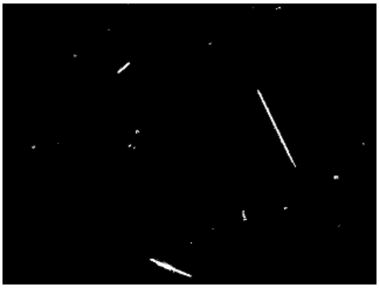


Fig. 3. Example of binarized difference image

of the binarized difference image is shown in Figure 3. The white pixels show large color difference in two images, we consider that asbestos are included in the white regions because asbestos have different colors in two images.

2.3 Asbestos Detector

An asbestos detector for classifying asbestos and other particles is trained using the local color features of two matched microscope images. As described above, the asbestos detector is developed using SVM of color features, and it is applied to only the pixels with large difference shown in Figure 3.

The color features for SVM are defined as

- RGB values of feature point (x, y) and eight neighbors of it in image 1 and 2 (9 pixels × 3 colors × 2 images).
- The average RGB values of local 3×3 regions whose center is (x, y) in image 1 and 2 (3 colors × 2 images).

The dimension of features is 60 $(= 9 \times 3 \times 2 + 3 \times 2)$. The asbestos detector is developed by SVM of the 60 dimensional color features. In the following experiments, a normalized polynomial kernel [9] is used as a kernel function. The kernel is defined as

$$K(\boldsymbol{x}, \boldsymbol{z}) = \frac{(\gamma\langle \boldsymbol{x} \cdot \boldsymbol{z}\rangle + r)^d}{\| (\gamma\langle \boldsymbol{x} \cdot \boldsymbol{x}\rangle + r)^d \| \cdot \| (\gamma\langle \boldsymbol{z} \cdot \boldsymbol{z}\rangle + r)^d \|} \tag{1}$$

where $\| \cdot \|$ represents norm and $\langle \cdot \rangle$ represents inner product. The parameters were set as $\gamma = 1$, $r = 1$ and $d = 5$ by preliminary experiment.

However, when the detector is applied to each pixel independently, there is the case that an asbestos particle is classified as two different particles. This increases false negatives. Moreover, there is also the case that non-asbestos particles and noise whose color is similar to asbestos are miss-classified as asbestos. This increases false positives. The reason is that the relation with neighboring regions is not used. To use the relation with neighboring regions, labels are estimated using CRF based on SVM outputs of each pixel. This corresponds to Support Vector Random Field (SVRF) [5].

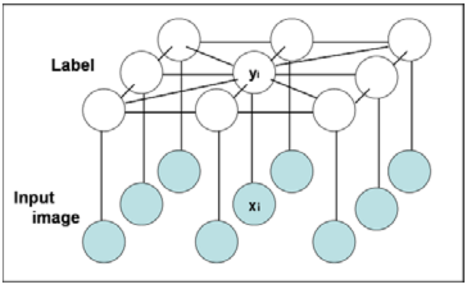


Fig. 4. CRF. The bottom nodes represent the pixels in the image X. The top nodes represent the label of the corresponding pixels. The edges represent the potential function.

CRF is succeeded in image labeling [2,3,5,8], and it models the posterior probability $P(Y|X)$ of the label set Y of the input image X directly. In CRF, posterior probability is defined as

$$P(Y|X) = \frac{1}{Z}\prod_{i\in S}\phi(y_i, X)\prod_{i\in S}\prod_{j\in N_i}\psi(y_i, y_j, X), \tag{2}$$

where ϕ and ψ are the potential functions. S is a set of a node i. In experiments, S corresponds to the number of pixels of an image. N_i is neighboring nodes of node i.

Figure 4 explains CRF. The bottom nodes in Figure 4 represent the pixels in the input image X. The top nodes represent the class labels of the corresponding pixels. $\phi(y_i, X)$ represents the relation of vertical direction. This is called as a observed potential. $\psi(y_i, y_j, x)$ represents the relation of horizontal direction. This is called as a local consistency potential. As shown in equation (2), labels are estimated using input image and neighboring labels.

2.4 Label Estimation by Support Vector Random Field

In this paper, the output of SVM at each pixel is used in the potential function $\phi(y_i, X)$. $\phi(y_i, X)$ using SVM output is defined as

$$\phi(y_i, X) = \frac{1}{1 + \exp(f(\Upsilon_i(X)))}, \tag{3}$$

where $\Upsilon_i(X)$ outputs the features of node i in image X. And f represents SVM decision function. The initial labels are determined by SVM as

$$y_i = \begin{cases} 1 \text{ (if } \operatorname{sgn}[f(\Upsilon_i(X))] = 1) \\ 0 \text{ (if } \operatorname{sgn}[f(\Upsilon_i(X))] = -1) \end{cases}. \tag{4}$$

The posterior probability is determined using the observation potential based on SVM output and local consistency potentials $\psi(y_i, y_j, x)$ as

$$P(Y|X) = \frac{1}{Z}\exp\left\{\sum_{i\in S}\log(\phi(y_i, f_i(X)) - \sum_{i\in S}\sum_{j\in N_i}\psi(y_i, y_j, X)\right\}, \tag{5}$$

$$Z = \sum_{Y}\exp\left\{\sum_{i\in S}\log(\phi(y_i, \varUpsilon_i(X)) - \sum_{i\in S}\sum_{j\in N_i}\psi(y_i, y_j, X)\right\}, \tag{6}$$

where Z is the normalization term. The local consistency potential is defined as

$$\psi(y_i, y_j, X) = \begin{cases} w_1 \text{ if } (y_i \neq y_j) \wedge (y_i = 1) \\ w_2 \text{ if } (y_i \neq y_j) \wedge (y_i = 0) \\ 0 \qquad\quad \text{otherwise} \end{cases} \tag{7}$$

Local-consistency potential should be small when label y_i and neighboring label y_j are the same label. On the other hand, the potential should be large when label y_i is different from label y_j. w_1 is used when the label y_i is asbestos and neighboring label y_j is not asbestos. Similarly, w_2 is used when label y_i is non-asbestos and y_j is asbestos. In this paper, local consistency potential is computed within a square region of 5×5 pixels. However, the area of asbestos within a square region is smaller than that of background because an asbestos particle is long and thin. Thus, if we set the same value to w_1 and w_2, asbestos labels tend to change to non-asbestos labels. To avoid this, w_2 is set to larger values than w_1. In the experiments, $w_1 = 0.1$ and $w_2 = 0.2$ are used. Potential function ψ in a equation (5) is based on the neighboring labels y_j. This means that the neighboring labels are required to estimate the label y_i. Thus, the computational cost becomes high. To reduce the cost, we use mean field approximation [6].

3 Experiments

The asbestos detector is applied to the while pixels in the binalized difference image shown in Figure 3. The white pixels have large difference in two images. First, we evaluate the accuracy of the detector based on only SVM. Next, the detection method based on SVRF of local color features is evaluated.

The 11 sets of microscope images are used to train the SVM. Each set consists of 2 microscope images captured with different angles of polarizing plate. The 21 asbestos particles are included in the 11 sets. The color features of 21 asbestos and many non-asbestos particles are used to train SVM. The 55 sets are used in evaluation. The 76 asbestos particles are included in the test images.

The evaluation result is shown in Table 1. The actual number is the number of the actual asbestos particles. 76 asbestos particles are included in test images. The number in the bracket is the true positive rate. False positive means that non-asbestos particles are mis-classified as asbestos. Since the detector uses only local color features, non-asbestos regions with similar color are miss-classified.

Table 1. Performance of the proposed method

	Actual number	True positive (Rate)	Number of false positives
SVM	76	54 (71.0%)	6
SVRF	76	60 (78.9%)	0

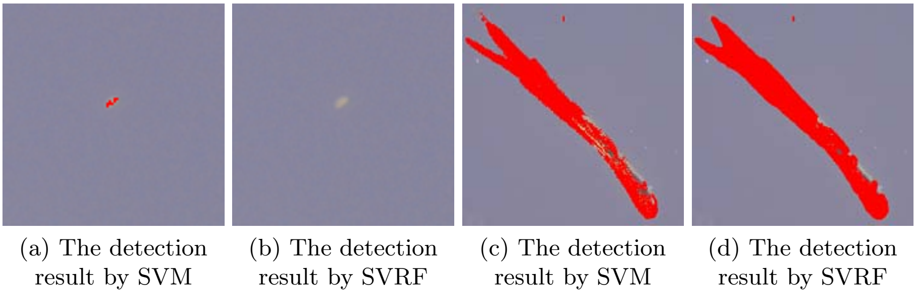

(a) The detection result by SVM (b) The detection result by SVRF (c) The detection result by SVM (d) The detection result by SVRF

Fig. 5. Examples of results improved by SVRF

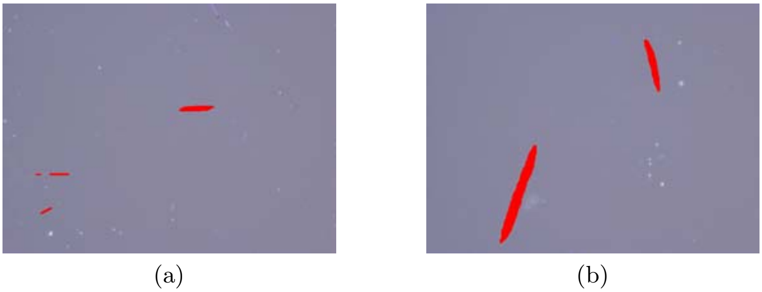

(a) (b)

Fig. 6. Example of detection result

This increases the false positives. Additionally, since the detector is applied to each pixel independently, there is the case that all pixels on an asbestos particle are not classified correctly.

Next, we evaluate the detector based on SVRF. Table 1 also shows the evaluation result. SVRF decreases false positives and increases true positives. Figure 5 (b) shows the example in which false positive is improved by SVRF. By using the relation with neighboring pixels, the false positive with small area is vanished. Figure 5 (d) shows the example in which false negative is improved by SVRF. Only the SVM based detector induces the case that an asbestos particle is detected as several asbestos shown in Figure 5 (c). This is because all pixels on an asbestos particle do not have the same color. By using SVRF, these asbestos regions are merged and detected as an asbestos particle correctly. However, SVRF can not recover all false negatives. For example, when the two regions with asbestos label are far, SVRF can not detect them as an asbestos particle. In this paper, we evaluate the accuracy strictly. The accuracy of the current version may not be sufficient. However, this research is a first step for automatic disperse dying method by computer. Since asbestos are used as building materials, this research will contribute to people's health in the world.

Finally, we show the examples of detection results in Figure 6. In Figure 6 (a), two asbestos are detected correctly while one asbestos at bottom left is recognized as two asbestos. Note again that we evaluate the accuracy strictly. This decreases true positive rate. In Figure 6 (b), all asbestos are detected correctly without false positives. These results show the possibilities of the proposed method.

4 Conclusion

In this paper, we propose the asbestos detection method in building materials. There are very few researches about asbestos detection in building materials. Since asbestos are used as building materials, this research will contribute to people's health in the world.

In experiments, two phase-contrast microscope images to which different polarization was applied were used. First, the position gap of two images was corrected, and SVM using local color features is applied to the pixels which have large difference in two images. Only SVM induces many false positives and negatives. Therefore, the labels are estimated by SVRF. Experimental results show that SVRF improves accuracy. The accuracy may not be sufficient but this is a giant step for automatic disperse dyeing method by computer.

The SVM based detector uses only local color features, and it can not give positive values to asbestos particles with unclear color. Since SVRF is based on the SVM outputs, it can not improve the case that SVM gives negative values to many pixels on an asbestos particle. To improve the accuracy further, we should use the shape features as well as color features. This is a subject for future works.

Acknowledgment

This research is supported by the research program (No.K1920 and No.K2061) of Ministry of the Environment of Japan.

References

1. Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
2. Toyoda, T., Hasegawa, O.: A random field model for integration of local information and global information. IEEE Trans. Pattern Analysis and Machine Intelligence 30(8), 1483–1489 (2008)
3. Wang, Y., Ji, Q.: A Dynamic Conditional Random Field Model for Object Segmentation in Image Sequences. In: Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 264–270 (2005)

4. Cristianini, N., Taylor, J.S.: An Introduction to Support Vector Machines: And Other Kernel-Based Learning Methods. Cambridge University Press, Cambridge (2000)
5. Lee, C.H., Greiner, R., Schmidt, M.: Support Vector Random Fields for Spatial Classification. In: Proc. European Conference on Principals and Practices of Knowledge Discovery in Data, pp. 121–132 (2005)
6. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proc. International Conference on Machine Learning, pp. 282–289 (2001)
7. Cortes, C., Vapnik, V.: Support-Vector Networks. Machine Learning 20, 273–297 (1995)
8. He, X., Zemel, R.S., Carreira-Perpinan, M.A.: Multiscale Conditional Random Fields for Image Labeling. IEEE CS Conference on Computer Vision and Pattern Recognition 2, 695–702 (2004)
9. Debnath, R., Takahashi, H.: Kernel selection for the support vector machine. IEICE Trans. on Information and System 87-D(12), 2903–2904 (2004)
10. Baron, P.A., Shulman, S.A.: Evaluation of the Magiscan Image Analyzer for Asbestos Fiber Counting. American Industrial Hygiene Association Journal 48, 39–46 (1987)
11. Kenny, L.C.: Asbestos Fiber Counting by Image Analysis - The Performance of The Manchester Asbestos Program on Magiscan. Annals of Occupational Hygiene 28(4), 401–415 (1984)
12. Inoue, Y., Kaga, A., Yamaguchi, K.: Development of An Automatic System for Counting Asbestos Fibers Using Image Processing. Particle Science and Technology 16, 263–279 (1989)

Acoustic Echo Cancellation Using Gaussian Processes

Jyun-ichiro Tomita[1] and Yuzo Hirai[2]

[1] Master's Program in Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan
[2] Department of Computer Science, University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan

Abstract. In this paper Gaussian process is applied to linear and nonlinear acoustic echo cancellation. Gaussian process is a kernel method in which predictions to new inputs are made based on the linear combination of kernel functions evaluated at each training data. First order acoustic echo-path is modeled by a linear equation of input data and second order acoustic echo-path is modeled by the second order polynomials. The performance of the cancellation is evaluated by white signal, stationary colored signal, non-stationary colored signal and real speech data. It is shown that more than 70 dB echo cancellation can be acieved within 400 ms.

1 Introduction

Acoustic echo cancellation is an indispensable component for a cellular phone to achieve smooth communication between speakers. Since the sizes of speaker and microphone become smaller and miniaturized, linear and nonlinear distortions in an acoustic echo path become larger.

A lot of works on echo cancellers using adaptive filters have been proposed so far [1]-[4]. It has been shown that an adaptive Volterra filter can eliminate nonlinear components effectively [5]-[8]. Volterra filter is a good nonlinear system model and can improve acoustic quality by suppressing nonlinear distortions. However, when the dimension of input becomes large and the degree of nonlinearity increases, the scale and the computational complexity of the filter also increase and they may degrade its convergence characteristics. Statistical nature of inputs also affects the performance of the filter.

Gaussian process is one of the stochastic processes that can be used to estimate an output value for a new input from learning data. It is a kernel method which assumes that the prior distribution over the parameter is a Gaussian distribution. The characteristic of its smoothness or periodicity is given by the kernel function. To define the kernel function corresponds to create a distribution over the function directly. In this method, the output value is estimated by the posterior distribution. Since this method does not use gradient information given by error signals, it does not suffer from ill conditions such as eigenvalue spread of a correlation matrix of input signals. Hence it is free from coloredness and stationarity of inputs as shown in the results.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 353–360, 2009.

2 Gaussian Processes

Gaussian process is defined as a probability distribution over functions $y(\boldsymbol{x})$ such that the set of values of $y(\boldsymbol{x})$ evaluated at a set of point $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N$ jointly have a Gaussian distribution. The joint distribution over N variables $y_1, \cdots, y_N$ is characterized by its mean and covariance.

2.1 Linear Regression

Consider a linear model that the output is given by a linear combination of M fixed basis functions denoted by a vector $\varPhi(\boldsymbol{x})$ so that

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \varPhi(\boldsymbol{x}) \tag{1}$$

where $\boldsymbol{x}$ is input vector and $\boldsymbol{w}$ is M-dimensional weight vector. Suppose a prior distribution over $\boldsymbol{w}$ is an isotropic Gaussian of the form

$$p(\boldsymbol{w}) = N(\boldsymbol{w}|\mathbf{0}, \alpha^{-1}\boldsymbol{I}) \tag{2}$$

where α denotes a hyperparameter which represents the precision of prior distribution.

The probability distribution over the weight vector $\boldsymbol{w}$ defined by Eq.(2) derives a probability distribution over functions $y(\boldsymbol{x})$. The output vector $\boldsymbol{y}$ is given by

$$\boldsymbol{y} = \boldsymbol{\varPhi}\boldsymbol{w} \tag{3}$$

where $\boldsymbol{\varPhi}$ is a design matrix with elements $\varPhi_{nk} = \phi_k(\boldsymbol{x}_n)$.

Since the output vector $\boldsymbol{y}$ is a linear combination of Gaussian distributed variables given by the elements of $\boldsymbol{w}$, the output $\boldsymbol{y}$ is also Gaussian. Its mean and covariance are given by

$$E[\boldsymbol{y}] = \boldsymbol{\varPhi}E[\boldsymbol{w}] \tag{4}$$

$$cov[\boldsymbol{y}] = E[\boldsymbol{y}\boldsymbol{y}^T] = \boldsymbol{\varPhi}E[\boldsymbol{w}\boldsymbol{w}^T]\boldsymbol{\varPhi}^T = \boldsymbol{K} \tag{5}$$

where $\boldsymbol{K}$ denotes the Gram matrix with elements

$$K_{nm} = k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \frac{1}{\alpha}\varPhi(\boldsymbol{x}_n)^T\varPhi(\boldsymbol{x}_m) \tag{6}$$

and $k(\boldsymbol{x}, \boldsymbol{x}^{'})$ is a kernel function.

2.2 Gaussian Processes for Regression

The joint distribution over the outputs $y_1, \cdots, y_N$ is the Gaussian distribution, so that the distribution is completely determined by its mean and covariance.

About the mean. There is no prior knowledge about the mean of $y(\boldsymbol{x})$. Hence, presume that the mean of $\boldsymbol{y}$ is zero. This is equivalent to choosing the mean of the prior distribution over weight values $p(\boldsymbol{w}|\alpha)$ to be zero.

About the covariance. The covariance function is given by the kernel function.

$$E[y(\boldsymbol{x}_n)y(\boldsymbol{x}_m)] = k(\boldsymbol{x}_n, \boldsymbol{x}_m) \tag{7}$$

We can define the kernel function directly. In this paper, we use linear kernel for both linear components and nonlinear parts. The linear kernel is given by

$$k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \boldsymbol{x}_n^T \boldsymbol{x}_m. \tag{8}$$

2.3 The Predictive Distribution

We need to consider the noise on the observed target values as follows

$$t_n = y_n + \epsilon_n \tag{9}$$

where $y_n = y(\boldsymbol{x}_n)$. The second term ϵ_n in the right hand side is a random noise variable which is given by a Gaussian distribution, so that

$$p(t_n|y_n) = N(t_n|y_n, \beta^{-1}) \tag{10}$$

where β is a hyperparameter and represents the precision of the noise. The joint distribution of the target values $\boldsymbol{t} = (t_1, \cdots, t_N)^T$ conditioned on the values of $\boldsymbol{y} = (y_1, \cdots, y_N)^T$ is given by a Gaussian.

$$p(\boldsymbol{t}|\boldsymbol{y}) = N(\boldsymbol{t}|\boldsymbol{y}, \beta^{-1}\boldsymbol{I}_N) \tag{11}$$

where $\boldsymbol{I}_N$ denotes an $N \times N$ unit matrix. In this conditions, the marginal distribution of $\boldsymbol{t}$ is given by

$$p(\boldsymbol{t}) = N(\boldsymbol{t}|\boldsymbol{0}, \boldsymbol{C}) \tag{12}$$

where the covariance matrix $\boldsymbol{C}$ has elements

$$C(\boldsymbol{x}_n, \boldsymbol{x}_m) = k(\boldsymbol{x}_n, \boldsymbol{x}_m) + \beta^{-1}\delta_{nm} \tag{13}$$

The joint distribution over $t_1, \cdots, t_{N+1}$ is given by

$$p(\boldsymbol{t}_{N+1}) = N(\boldsymbol{t}_{N+1}|\boldsymbol{0}, \boldsymbol{C}_{N+1}) \tag{14}$$

where $\boldsymbol{C}_{N+1}$ is an $(N+1) \times (N+1)$ covariance matrix with elements given by Eq.(13). To consider the conditional distribution $p(t_{N+1}|\boldsymbol{t}_N)$, we partition the covariance matrix as follows

$$\boldsymbol{C}_{N+1} = \begin{pmatrix} \boldsymbol{C}_N & \boldsymbol{k} \\ \boldsymbol{k}^T & c \end{pmatrix} \tag{15}$$

where $\boldsymbol{k} = k(\boldsymbol{x}_n, \boldsymbol{x}_{N+1})$ for $n = 1, \ldots, N$, and the scalar $c = k(\boldsymbol{x}_{N+1}, \boldsymbol{x}_{N+1}) + \beta^{-1}$. The conditional distribution $p(t_{N+1}|\boldsymbol{t}_N)$ is a Gaussian distribution, and its mean and covariance are given by

$$m(\boldsymbol{x}_{N+1}) = \boldsymbol{k}^T \boldsymbol{C}_N^{-1} \boldsymbol{t} \tag{16}$$

$$\sigma^2(\boldsymbol{x}_{N+1}) = c - \boldsymbol{k}^T \boldsymbol{C}_N^{-1} \boldsymbol{k} \tag{17}$$

The vector $\boldsymbol{k}$ is a function of the input vector $\boldsymbol{x}_{N+1}$. Therefore, the predictive distribution is a Gaussian distribution whose mean and variance both depend on the input vector $\boldsymbol{x}_{N+1}$. That is, the mean and the variance of the output value y_{N+1} are determined by the new input vector $\boldsymbol{x}_{N+1}$.

3 Proposed Method

3.1 The Echo Canceller Using Gaussian Processes

We propose a new method which cancels acoustic echo by subtracting spurious echo $y(n)$ which is calculated by Gaussian processes from the desired response (the echo) $d(n)$, as illustrated in Figure 1. Since the desired response includes linear components and nonlinear components, the input vector for Gaussian processes must involve elements for both of them. Thus, we make higher order components from the input vector. We assume that the degree of nonlinearity is second order.

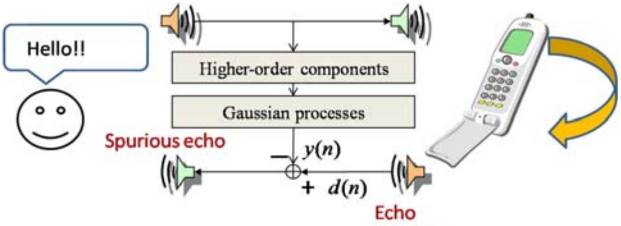


Fig. 1. Acoustic echo canceller using Gaussian processes

3.2 Updating of the Covariance Matrix

The newest N samples of the learning data are used for an $N \times N$ covariance matrix, so that elements of the covariance matrix are generated by N samples. Therefore, the estimated value is given after Nth sample. The estimated value at the nth sample ($n > N$) is given by the nth input data $\boldsymbol{x}_n$, desired responses $\boldsymbol{d}_{n-N}, \cdots, \boldsymbol{d}_{n-1}$ and the covariance matrix generated by $\boldsymbol{x}_{n-N}, \cdots, \boldsymbol{x}_{n-1}$. Then we delete elements corresponding to the oldest learning data from the covariance matrix, and update the covariance matrix by using the nth input data $\boldsymbol{x}_n$, as illustrated in Figure 2. Estimation of the output value corresponds to obtaining the mean value $m(\boldsymbol{x}_n)$. Hence, we take the mean value as the spurious echo.

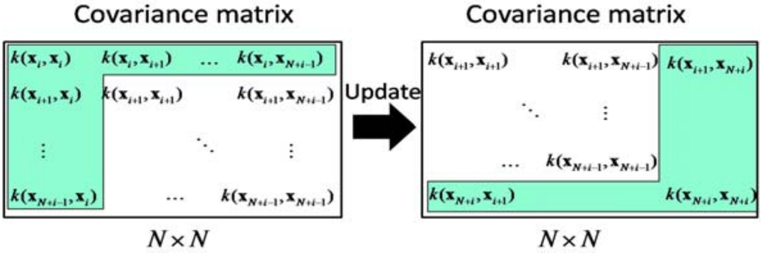

Covariance matrix
$k(\mathbf{x}_i,\mathbf{x}_i)$
$k(\mathbf{x}_i,\mathbf{x}_{i+1})$
$\cdots$
$k(\mathbf{x}_i,\mathbf{x}_{N+i-1})$
$k(\mathbf{x}_{i+1},\mathbf{x}_i)$
$k(\mathbf{x}_{i+1},\mathbf{x}_{i+1})$
$k(\mathbf{x}_{i+1},\mathbf{x}_{N+i-1})$
$\vdots$
$\ddots$
$\vdots$
$k(\mathbf{x}_{N+i-1},\mathbf{x}_i)$
$\cdots$
$k(\mathbf{x}_{N+i-1},\mathbf{x}_{N+i-1})$
$N \times N$
Update
Covariance matrix
$k(\mathbf{x}_{i+1},\mathbf{x}_{i+1})$
$k(\mathbf{x}_{i+1},\mathbf{x}_{N+i-1})$
$k(\mathbf{x}_{i+1},\mathbf{x}_{N+i})$
$\ddots$
$\vdots$
$\cdots$
$k(\mathbf{x}_{N+i-1},\mathbf{x}_{N+i-1})$
$k(\mathbf{x}_{N+i},\mathbf{x}_{i+1})$
$k(\mathbf{x}_{N+i},\mathbf{x}_{N+i})$
$N \times N$

Fig. 2. Updating covariance matrix

Table 1. Simulation conditions

	Proposed method	Adaptive Volterra filter
Linear acoustic echo-path N_L	50	
Nonlinear acoustic echo-path N_{NL}	50×50	
Size of covariance matrix	2500×2500	-
Learning method	-	RLS method ($\gamma = 1$)
Input signal	White signal,Stationary colored signal Non-stationary colored signal,Real speech data	
Sampling frequency	8kHz	
Time	5 second	

Table 2. Parameters of the second order AR circuit

Parameter	Stationary colored signal	Non-stationary colored signal
r	0.9	0.9
θ	0	$\cos\left(\frac{2\pi n}{L}\right), L = 50000$
ω	$\frac{\pi}{2}$	$\frac{\pi}{2}$

4 Simulation Results

We have simulated the proposed method for white signal, stationary signal, non-stationary signal and real speech data which is spoken by a man. For comparison adaptive Volterra filter was also simulated with the same set of inputs. Table 1 shows conditions of simulation, and the linear and the nonlinear acoustic echo-path are illustrated in Figures 3 and 4 respectively.

Stationary colored signal and non-stationary colored signal are generated by a second order AR circuit which is illustrated in Figure 5. We can generate both stationary colored signal and non-stationary colored signal by entering white signal to the second order AR circuit. Table 2 shows parameters of the second order AR circuit. We introduce non-stationarity by moving the poles according to cosine function as illustrated in figure 6. We set the size of covariance matrix to 2500 × 2500 empirically.

4.1 Measure of Echo Suppression

Usually, speaking in a room generates echo but we cannot notice it because the delay between the actual sound and the echo at our ears is small. As the delay becomes large we can notice echo and it becomes difficult to maintain smooth conversation. Therefore suppression of acoustic echo in cellular phone is necessary because usually delay is large in telephone lines. The Echo Return Loss Enhancement (ERLE) is used to specify required suppression level of echo to maintain smooth conversation. The ERLE is defined by

$$ERLE[dB] = 10 \log_{10} \frac{\sum \{d(n)\}^2}{\sum \{d(n) - y(n)\}^2} \tag{18}$$

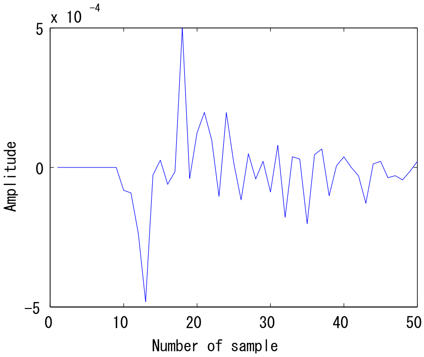


Fig. 3. Linear acoustic echo-path

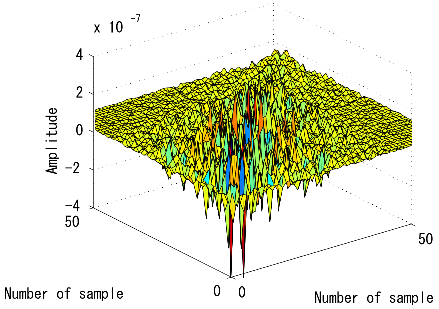


Fig. 4. Nonlinear acoustic echo-path

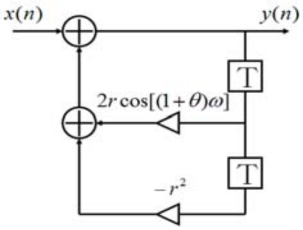


Fig. 5. The second order AR circuit

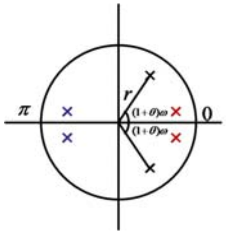


Fig. 6. The poles of the signal which are moved to produce nonstationarity

where $d(n)$ denotes desired response and $y(n)$ denotes estimated output.

According to [9], the required ERLE is over 50 dB when the delay time is more than 300 ms. In this paper, we suppose that the desired ERLE is 50 dB.

4.2 Simulation Results

The performance of the proposed method is evaluated by a series of simulation studies. The left hand side of Figure 7 shows the time variations of ERLEs for adaptive Volterra filter using the RLS method for the suppression of echoes of the following inputs: white signal, stationary colored signal, non-stationary colored signal and real speech signal. The right hand side of Figure 7 shows the results for our proposed method to the same inputs.

By this method, the ERLE of each input converges over 70 dB (more than the desired ERLE of 50 dB) within 320 milliseconds. The initial convergence ERLEs are better than those obtained by Adaptive Volterra filter using RLS method. Moreover, the initial convergence characteristics of ERLE is not affected by statistical natures of inputs such as coloredness and nonstationarity. For stationary signals, the proposed method shows better convergence speed than the Volterra filter with RLS method. For nonstationary signals including actual

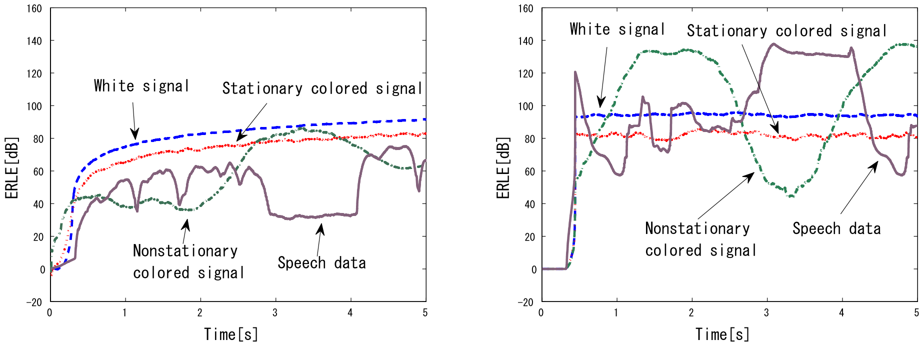


Fig. 7. Result for adaptive Volterra filter using the RLS method (left) and for Gaussian processes using linear kernel (right)

speech data, the proposed method shows better ERLEs at almost every period after initial convergence.

5 Conclusion

Gaussian process is applied to acoustic echo cancellation. It is shown that the echo which involves both linear components and nonlinear components can be cancelled more than 70 dB. The initial convergence speeds for statistically different inputs including real speech signal are almost identical as opposed to previously proposed algorithms.

In our method, since the inverse of covariance matrix must be calculated, it is difficult to run the algorithm in real-time especially when we consider nonlinear components. We are looking for a way to reduce computation time.

We are also looking for a way to solve both double talk problem and noise problem. When both speakers at their phones talk at the same time, the echo signal of one speaker overlaps with the talk signal of another speaker and clean learning signal for the echo cancellation cannot be obtained. Moreover, we cannot catch clean learning signal when the noise signal becomes larger. Detection of both double talk and noise signal and to stop learning during those periods will be a possible solution.

The precision of estimation depends on kernel function. In this study, we use a linear kernel for kernel function. We will study other kernel functions such as exponential kernel and Gaussian kernel for improving the performance.

References

1. Strenger, A., Kellermann, W.: Adaptation of a memoryless preprocessor for nonlinear acoustic echo cancelling. Signal Processing 80, 1747–1760 (2000)
2. Messerschmidt, D.: Echo cancellation in speech and data transmission. IEEE J. Select. Areas Commun. SAC-2, 283–297 (1984)

3. Strenger, A., Trautmann, L., Rabenstein, R.: Nonlinear Acoustic Echo Cancellation with 2nd Order Adaptive Volterra Filters. In: IEEE. Proc. ICASSP 1999, pp. 877–880 (1999)
4. Nakayama, K., Hirano, A., Kashimoto, H.: A Lattice Predictor Based Adaptive Volterra Filter and Its Convergence Property Analysis. IEICE Technical Report SIP2004-3 (2004)
5. Koh, T., Powers, E.J.: Second-order Volterra filtering and its application to nonlinear system identification. IEEE Trans. Acoustic, Speech, and Signal Processing 33(6), 1445–1455 (1985)
6. Haykin, S.: Adaptive Filter Theory, 4th edn. Prentice-Hall, Englewood Cliffs (2001)
7. Farhang-Boroujeny, B.: Adaptive Filters, Theory and Applications. John Wiley and Sons, Chichester (1998)
8. Schetzen, M.: The Volterra and Wiener theories of nonliner systems, Kriger (1989)
9. Yasukawa, H., Ogawa, M., Nishino, M.: Echo Return Loss Required for Acoustic Echo Controller Based on Subjective Assessment. IEICE Transactions E 74(4) (April 1991)
10. Yang, J.-M., Sakai, H.: A New Adaptive Algorithm for Echo Cancellation Using Indepent Component Analysis, IEICE Technical Report EA2006-17 (2006)
11. Bishop, C.M.: Pattern recognition and machine lerning. Springer, Heidelberg (2006)
12. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Lerning. MIT Press, Cambridge (2006)
13. Gibbs, M., MacKay, D.J.C.: Efficient Implementation of Gaussian Process, Technical report, Cavendish Laboratory, Cambridge, UK (1997)
14. Rasmussen, C.E.: Evaluation of Gaussian processes and other methods for non-Linear regression, PhD Thesis. University of Toronto, Department of Computer Science, Tronto, Canada (1996)
15. Oba, S., Ishii, S., Sato, M.: On-Line Learning Methods for Gaussian Processes. IEICE transactons on information and systems E 86-D(3), 650–654 (2003)

Automatic Particle Detection and Counting by One-Class SVM from Microscope Image

Hinata Kuba, Kazuhiro Hotta, and Haruhisa Takahashi

The University of Electro-Communications,
1-5-1 Chofugaoka, Chofu, Tokyo 182-8585, Japan
{kuba,hotta,Takahasi}@ice.uec.ac.jp

Abstract. Asbestos-related illnesses become a nationwide problem in Japan. Now human inspectors check whether asbestos is contained in building material or not. To judge whether the specimen contains asbestos or not, 3,000 particles must be counted from microscope images. This is a major labor-intensive bottleneck. In this paper, we propose an automatic particle counting method for automatic judgement system whether the specimen is hazardous or not. However, the size, shape and color of particles are not constant. Therefore, it is difficult to model the particle class. On the other hand, the non-particle class is not varied much. In addition, the area of non-particles is wider than that of particles. Thus, we use One-Class Support Vector Machine (OCSVM). OCSVM identifies "outlier" from input samples. Namely, we model the non-particle class to detect the particle class as outlier. In experiments, the proposed method gives higher accuracy and smaller number of false positives than a preliminary method of our project.

1 Introduction

Asbestos-related illnesses become a nationwide problem in Japan. Asbestos was widely used as building materials in Japan after the high-growth period of the 1970s. However, the use of asbestos has been banned or limited worldwide since the late 1980s, because it was discovered to cause cancer. Therefore, we need to check whether asbestos is used or not in a building when the building is demolished or renovated.

There are some automatic airborne asbestos detection methods [1,2,3]. In this paper, we pay attention to asbestos detection problem in building materials not atmosphere. Asbestos detection in building materials is more difficult than that in atmosphere, because the specimen from building materials includes various kinds of particles. In near future, asbestos detection problem in building materials will become important more and more. The analysis by human inspectors is called as disperse dyeing method. Disperse dyeing method prepares three specimens from one sample, and 1,000 particles are counted from each specimen. When more than four asbestos are contained in 3,000 particles, it is judged as dangerous. The detection of individual particles from microscope images is a

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 361–368, 2009.

major labor-intensive bottleneck for human inspectors. Thus, we propose an automatic particle detection and counting method to realize the disperse dyeing method by computer.

The particle detection problem is a binary classification between the particle class and the non-particle class. However, the size, shape and color of particles are not constant. Therefore, it is difficult to model the particle class. On the other hand, the non-particle class is not varied much. In addition, the area of non-particles is wider than that of particles. Thus, we model the non-particle class to detect the particles as "outlier". For this purpose, we use One-class Support Vector Machine (OCSVM) [4,5,6] for outlier detection.

In the experiments, we use a background image as a preprocessing. The background image is captured without the specimen. We compute the difference between an input image and the background image. We expect that regions with large difference contain particles. After computing the difference image, the proposed method is applied to the difference image, and particles are detected and counted. The proposed method has two steps. The first step is the detection of particle regions from the difference gray scale image by OCSVM. The gray scale image is divided into small regions without overlap, and we make a gray level histogram of each region. The histograms are used as input features for OCSVM. OCSVM determines that particles exist or not into small regions. The second step is counting particles. It is expected that the non-particle regions contain only the background. On the other hand, the regions which are classified as particle class may contain the particle and background, because we assign the label (particle or non-particle) to the square regions by OCSVM. Thus, we must eliminate the background from the regions with the particles label. We binarize the difference image using the higher and lower intensity values of regions with the non-particle label as a threshold. By this binarization, the background is eliminated from the regions with the particle label. Then the number of particles is counted.

The 20 microscope images which include the particles and other particles offered by RIKEN are used. We compared the proposed method with a preliminary particle detection method of our project. The preliminary method is based on the binarization of the difference image between an input and background image. The proposed method gives higher accuracy and smaller number of false positives than the preliminary method.

The remainder of this paper is organized as follows. Section 2 covers the particle detection and counting by OCSVM. Section 3 gives the experimental results, followed by conclusions in the last section.

2 Particle Detection and Counting by One-Class SVM

In this paper, we propose an automatic particle detection and counting method based on OCSVM from microscope images. Figure 1 shows the examples of particles in microscope images of building materials. We understand that the size, shape and color of particles are not constant. It is difficult to model the

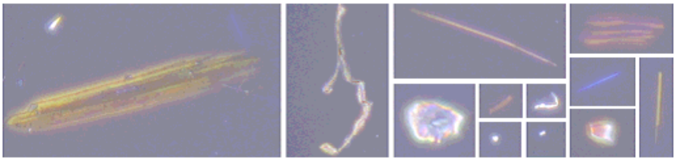


Fig. 1. Example of particles in building materials

particle class. On the other hand, the non-particle class is not varied much. In addition, the area of non-particles is wider than that of particles as shown in Figure 4. Thus, we detect particles as "outlier" by OCSVM. The OCSVM is the unsupervised learning for outlier detection. Although the SVM classifier needs to train in advance, the OCSVM does not need. Therefore, the particle detection by OCSVM is appropriate for practical system. Only one thing to do is to capture a background image at the current environment. This is an advantage of our method.

In section 2.1, we explain OCSVM. Section 2.2 describes the proposed particle detection and counting method.

2.1 One-Class SVM

OCSVM is the method for detecting outliers from input samples. The OCSVM maps an input sample into a high dimensional feature space F via a non-linear mapping Φ and finds the maximum margin hyperplane which separates the input sample between the origin and desired hyperplane. Figure 2 shows the outlier detection by OCSVM. The 3 samples which are near to the origin are the outliers in this example.

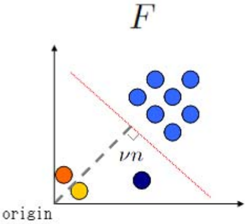


Fig. 2. Outlier detection by OCSVM

Let $\mathbf{x_1}, \ldots, \mathbf{x_n} \in \mathbf{X}$ are input vectors. In order to separate the data from the origin, it needs to solve the optimization problem:

$$minimize : \frac{1}{2}||\boldsymbol{w}||^2 + \tfrac{1}{\nu n} \sum_{i=1}^{n} \xi_i - \rho,$$

$$subject\ to : \quad \langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle \geq \rho - \ \xi_i \ \ , \quad \xi_i \geq 0, \quad i \in [n],$$

where ϕ is the non-linear transform, $\boldsymbol{w}$ and ρ denote a weight vector and a threshold. ξ_i are slack variables that penalize the objective function with allowing some of the feature vectors to be located in between the origin and desired hyperplane. $\nu \in (0,1)$ is the trade-off parameter that controls between the margin and the penalty. $\nu = 0$ means OCSVM with hard margin.

The decision function $f(\mathbf{z})$ by OCSVM is defined as

$$f(\mathbf{z}) = sgn(\langle \boldsymbol{w}, \phi(\boldsymbol{x}) \rangle - \rho)),$$

The dual problem of the optimization problem defined as

$$minimize: \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j),$$

$$subject\ to: \quad 0 \leq \alpha \leq \frac{1}{\nu n}, \quad \sum_{i=1}^{n} \alpha_i = 1,$$

where α are Lagrange multipliers and K is the kernel function.

In OCSVM, Gaussian kernel is frequently used to map outliers near to the origin. We also use the Gaussian kernel defined as

$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\frac{||\boldsymbol{x} - \boldsymbol{y}||^2}{\sigma^2}),$$

where σ denotes the standard deviation.

In this paper, we used the LIBSVM [7]. The parameters are set as $\nu = 0.9, \gamma = 1, c = 1$ by preliminary experiment.

2.2 Particle Detection and Counting Method

As a preprocessing, we compute the difference between an input image and a background image. The background image does not include any particles. It was captured without specimen. Although both the input and background images are color, the difference image is the gray scale by computing the squared distance of RGB signals of each pixel. The size of the difference image is equal to the input image. The proposed method has two steps. First, the particle regions are detected roughly by OCSVM. Second, particles are counted. Figure 3 shows how to apply OCSVM. In the first step, the difference image is divided into small regions without overlap. The size of small region is set to 10 × 10 pixels. Therefore, when the size of an input image is 100 × 100 pixels, 100 local regions are obtained. In this paper, we use histogram as a feature of each region because the position of particles in a small region is not constant. The histograms of all regions are fed into OCSVM, and the outliers (particles) are selected from 100 regions by OCSVM.

Figure 4-9 shows the flow of our approach. Figure 4 is an input image and Figure 5 is a background image. In this paper, only one background image shown in Figure 5 is used in the experiments. Figure 6 shows the difference image. The difference image becomes grayscale by computing the squared distance of RGB colors at each pixel. Particles have large difference values. We compute the histograms of small regions of Figure 6, and they are fed into OCSVM. Then the

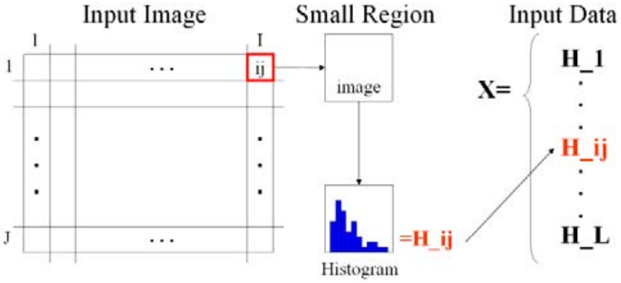


Fig. 3. How to apply OCSVM. When the size of an input image is 100×100 pixels and the size of a small region is set to 10×10 pixels ($I = 10, J = 10$), 100 local regions are obtained ($L = I \times J = 100$).

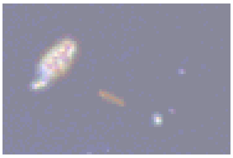


Fig. 4. Original image

Fig. 5. Background image

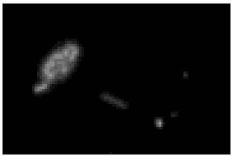


Fig. 6. Difference image

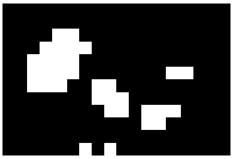


Fig. 7. Result by OCSVM

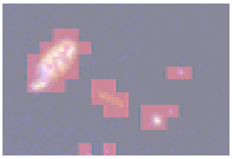


Fig. 8. Result by OCSVM on original image

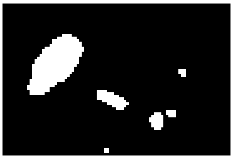


Fig. 9. Final result

result shown in Figure 7 is obtained. White regions show the particle label and black regions show the non-particle label. Since squared local regions are used as input samples for OCSVM, the result like the block is obtained. In Figure 8, the regions with the particle label which are shown as pink are overlapped to Figure 4. Since OCSVM assigns labels to squared regions, the regions with the particle label in Figure 8 contain both particles and background. We want to eliminate this for counting particles correctly. To eliminate it, we use the information of the regions classified as background. We expect that the regions with the non-particle label contain only background. Thus, we use the higher and lower intensity values of the regions with non-particle label as a threshold. The result is shown in Figure 9. It can be seen that the result of Figure 9 is more precise than that of Figure 7. We count the particles by labeling from the Figure 9.

3 Experiments

The 20 microscope images which include asbestos and other particles offered by RIKEN[11] are used. In the 20 images, 1,051 particles are included. The size of the microscope images is 640×480 pixels. These are color images. As a preprocessing, the difference grayscale image is computed. To obtain the input features for OCSVM, the difference image is divided into the small regions of 10×10 pixels without overlap. Then we make the histogram of each region. The histogram bin size is set to 30.

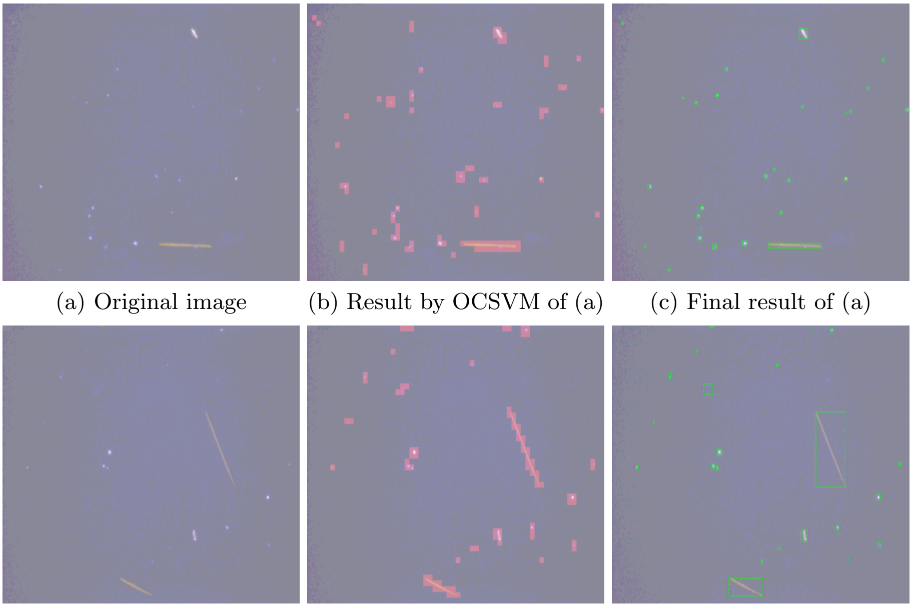
To show the effectiveness of the proposed method, we compared it with a preliminary method of our project. The preliminary method has 3 steps. First, the difference image between the input and background image is computed. Second, it is binarized by Otsu's binarized method [8]. Third, particles are counted by labeling in the binarized image. We evaluate the methods by using the true positive rate and the number of false positives. The true positive means that particles are classified correctly. The false positive means that non-particles are mis-classified as the particle class. Table 1 shows the results of our method and the preliminary method. True positive rate of our method achieves 88 % while that of the comparison method achieves only 58 %. In addition, the number of false positive is smaller than the comparison method. These results show the effectiveness of particle detection by OCSVM. Figure 10 shows the examples of particle detection by our method.

Figure 10 (a) and (d) show the original input images. Figure 10 (b) and (e) show the results by OCSVM of (a) and (d). The background is already eliminated by the binarization. The regions with particle label are shown as pink. Figure 10 (c) and (f) show the final result of our method. The green rectangle shows the one particle. The proposed method can detect particles with various sizes correctly. This is a giant step to realize the automatic disperse dyeing method by computer, because there are few researches about asbestos detection in building materials by computer.

Since the proposed method is based on the difference image, it failed to detect particles which have the close value to background. Almost of all false negatives are this kind of error. In terms of the false positives, there are some error types. The typical false positives are shown in Figure 11. Figure 11 (a) is air bubble which is not a particle. In our method, the regions with large difference from background are detected as particles. Thus, air bubbles are also detected. We count them as false positives strictly. Figure 11 (b) is the example of overlap of some particles. Our method detect it as one particle. It may need to the stereoscopic system. Figure 11 (c) shows the example of out of focus. The microscope has shallow depth of field. Then the captured image come into focus, or out of focus by location. The particles with out of focus are smoothed and have the close value of background. Therefore, it failed to detect particle. Figure 11 (d) is the example of a fiber with broken pieces. Human inspector can count this as one particle, however the proposed method count it as some particles. We judge them as false positives strictly. In this paper, the simple labeling algorithm is used but this will be improved by using conditional random field [9].

Table 1. Evaluation results

	Number of particles	True positive rate	Number of false positive
Proposed method	1,051	88(%)	74
Preliminary method	1,051	58(%)	126


(a) Original image (b) Result by OCSVM of (a) (c) Final result of (a)

(d) Original image (e) Result by OCSVM of (d) (f) Final result of (d)

Fig. 10. Examples of particle detection by the proposed method. (a) and (d) Original images. (b) and (e) The regions with particle label which are shown as pink are overlapped to the original images. (c) and (f) The final results.

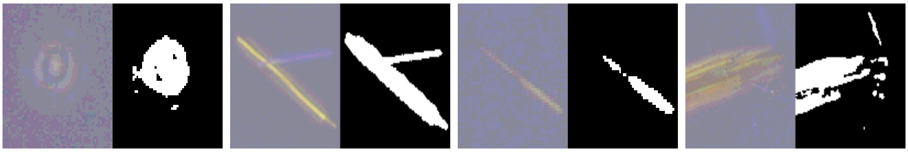

(a) (b) (c) (d)

Fig. 11. Example of false positive. (a) an air bubble (b) an overlap of particles (c) a decoupling by the out of focus (d) a fiber with broken pieces

4 Conclusion

In this paper, we propose an automatic particle counting method using OCSVM for automatic disperse dyeing method from building materials by computer. We detect particles from the microscope image as outliers by OCSVM. Experimental results show the effectiveness of the proposed the particle detection method. Although we use the histogram as features for OCSVM,the Gaussian kernel is used in OCSVM. The kernel specialized for histogram may improve the accuracy futher. The pyramid match kernel [10] may be used for this purpose. This is a subject for future works.

Acknowledgements

This research is supported by the research program (No.K1920 and No.K2061) of Ministry of the Environment of Japan.

References

1. Baron, P.A., Shulman, S.A.: Evaluation of the Magiscan Image Analyzer for Asbestos Fiber Counting. American Industrial Hygiene Association Journal 48, 39–46 (1987)
2. Kenny, L.C.: Asbestos Fiber Counting by Image Analysis - The Performance of The Manchester Asbestos Program on Magiscan. Annals of Occupational Hygiene 28(4), 401–415 (1984)
3. Inoue, Y., Kaga, A., Yamaguchi, K.: Development of An Automatic System for Counting Asbestos Fibers Using Image Processing. Particle Science and Technology 16, 263–279 (1989)
4. Schölkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C.: Estimating the Support of a High-Dimensional Distribution. Neural Computation 13(7), 1443–1471 (2001)
5. Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons, Chichester (1998)
6. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
7. Chang, C.C., Lin, C.J.: LIBSVM (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm
8. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Trans. on System, Man, and Cybernetics SMC-9(1), 62–66 (1979)
9. Lafferty, J., McCallum, A., Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: Proceedings of the Eighteenth International Conference on Machine Learning, ICML (2001)
10. Grauman, K., Darrell, T.: The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), Beijing, China (October 2005)
11. http://www.riken.jp/engn/index.html

Protein Folding Classification by Committee SVM Array

Mika Takata and Yasuo Matsuyama

Department of Computer Science and Engineering,
Waseda University, Tokyo 169-8555, Japan
{m-kan_515_hope,yasuo}@wiz.cs.waseda.ac.jp
http://www.wiz.cs.waseda.ac.jp/index-e.html

Abstract. Protein folding classification is a meaningful step to improve analysis of the whole structures. We have designed committee Support Vector Machines (committee SVMs) and their array (committee SVM array) for the prediction of the folding classes. Learning and test data are amino acid sequences drawn from SCOP (Structure Classification Of Protein database). The classification category is compatible with the SCOP. SVMs and committee SVMs are designed in an one-versus-others style both for chemical data and sliding window patterns (spectrum kernels). This generates the committee SVM array. Classification performances are measured in view of the Receiver Operating Characteristic curves (ROC). Superiority of the committee SVM array to existing prediction methods is obtained through extensive experiments to compute the ROCs.

1 Introduction

Learning machines have been presenting powerful methods to bioinformaticians. Some of early tools already appear in monographs [1] and are accessible through the Internet. Recent advancement of the learning algorithm is well-supported by the enforcement of computing power. Among many bioinformatics fields, the prediction[1] of various structures from given {DNA, amino acid}-sequences is considered to be an attractive target from the learning theory viewpoint. This paper is compatible with this stance.

The protein function is strongly related to its folding pattern. But, there are unknown structures overwhelmingly more than known ones which are assembled in databases. Therefore, the prediction of the protein folding class *in silico* helps wet biologists greatly so that haphazard experiments can be avoided. Therefore, this paper addresses the protein folding classification using computationally learned machines. The SCOP (Structure Classification Of Protein database) [2] is the resource of training data, test data, and classification categories. Methods presented in this paper are as follows.

[1] In bioinformatics, this terminology is equivalently called recognition, estimation, classification, etc.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 369–377, 2009.

(1) SVMs (Support Vector Machines) on data reflecting chemical properties,
(2) SVMs on N-grams,
(3) A committee SVM array on chemical data,
(4) A committee SVM array on chemical data and N-grams.

All SVMs in this paper are designed in an one-versus-others style. This leads to the array structure of (3) and (4). Each performance is evaluated by a Receiver Operating Characteristic curve (ROC). Since data entities of (1) and (2) are very different, the committee SVM arrays of (3) and (4) improve the classification performance. Experiments on the protein folding classification show that the committee SVM array of (4) has a superior performance to existing ones [3], [4], [5], [6], [7] by revealing a balance at around (specificity, sensitivity)=(85.40%, 76.13%).

2 Protein Folding Classes, Feature Parameters, and Structure Descriptors

2.1 SCOP Folding Classes

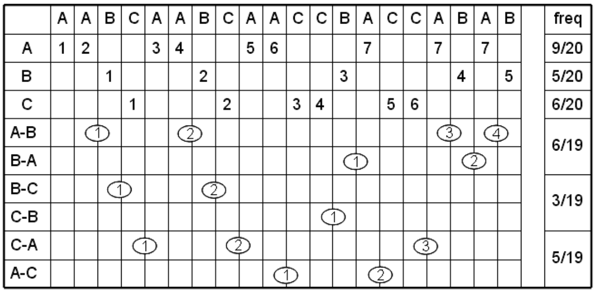
SCOP data set [2] contains 27 fold classes under 4 super families; α (6 types of folds), β (9 types of folds), α/β (9 types of folds), and $\alpha + \beta$ (3 types of folds). These names reflect the α-helix and the β-sheet of protein structures. The 27 fold types are indexed by $\{1, 3, 4, 7, 9, 11, 20, 23, 26, 30, 31, 32, 33, 35, 39, 46, 47, 48, 51, 54, 57, 59, 62, 69, 72, 87, 110\}$. On the list of the 27 types of protein folds, readers are requested to refer to [5] since what we need are identifiers alone.

2.2 Feature Vectors

There are several methods for the extraction of feature vectors which are fed into a learning machine. Reasonable extraction of feature vectors leads to a reliable performance on the fold classification.

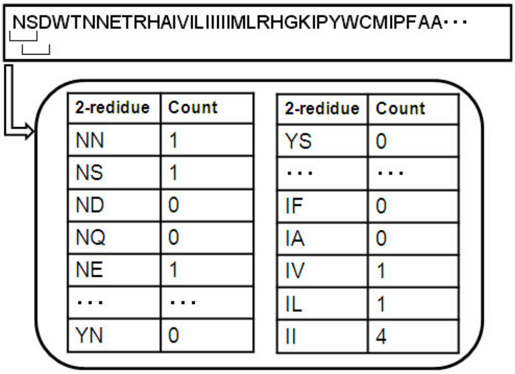
Global Protein Structure Descriptor: For a sequence with 3-letter alphabets, there is a method to generate a 21 dimensional feature vector. Another protein structure descriptor is developed in the day of [3]. In this method, each residue (amino acid symbol) is classified to types {A, B, C} according to the amino acid attribute [4]. Then, frequencies of X and transitions from X to Y (denoted by X-Y) or Y to X (denoted by Y-X) are computed as is illustrated in Fig. 1. Here, X and Y are one of {A, B, C} with X≠Y. The last column of Fig. 1 shows the frequencies giving a 6-dimensional sub-vector. Furthermore, {the first letter, 25% frequency, 50% frequency, 75% frequency, 100% frequency} makes a 5-dimensional vector. Therefore, a 21-dimensional feature vector (21=6+5×3) is obtained from each attribute.

CSHVPZ Vectors: Input vectors to SVMs are made from an amino acid sequence as follows [5].



	A	A	B	C	A	A	B	C	A	A	C	C	B	A	C	C	A	B	A	B		freq
A	1	2			3	4			5	6				7			7		7			9/20
B			1				2						3					4		5		5/20
C				1				2			3	4			5	6						6/20
A-B		①				②											③		④			6/19
B-A													①					②				
B-C			①				②															3/19
C-B												①										
C-A				①				②								③						5/19
A-C										①				②								

Fig. 1. Global protein structure descriptor



NSDWTNNETRHAIVILIIIIIMLRHGKIPYWCMIPFAA···

2-redidue	Count
NN	1
NS	1
ND	0
NQ	0
NE	1
···	···
YN	0

2-redidue	Count
YS	0
···	···
IF	0
IA	0
IV	1
IL	1
II	4

Fig. 2. Sliding window frequency

Symbol C : 20-dimensional vector is is obtained from frequencies of 20 amino acid symbols.

Symbol S : 21-dimensional vector is obtained by from three types of secondary structure prediction.

Symbol H : 21-dimensional vector is obtained from the hydrophobicity.

Symbol V : 21-dimensional vector is obtained from the normalized van der Waals volume.

Symbol P : 21-dimensional vector is obtained from the polarity.

Symbol Z : 21-dimensional vector is obtained from the polarizability.

We call each of the above data C,S,H,V,P,Z vectors. If all feature vectors are piled up, a 125-dimensional super vector is obtained. We call this data CSHVPZ super vector.

2.3 Sliding Window N-Gram

A sliding window N-gram extracts a global pattern in a discrete symbol sequence.

$$c\,(\sigma, \mathbf{x}) = (number\ of\ occurrence\ of\ \sigma\ in\ \mathbf{x})/(length\ of\ \mathbf{x}). \tag{1}$$

Here, σ is an N-fold element in Σ^N. In this paper, Σ is the set of 20 amino acid symbols. From now on, k will be used for the length instead of N. Figure 2 illustrates the counting procedure of σ for $k = 2$. In later experiments, the cases of $k = 2$ and $k = 3$ will be examined. Note that $k = 3$ reveals 8000-dimensional vector data to be classified, which is a practical limit due to the machine learning complexity.

The function $c(\sigma, \mathbf{x})$ is used to derive the spectrum kernel [6] or the string kernel [8]. Spectrum kernel elements are defined as follows.

$$K\left(\mathbf{x}, \mathbf{x}'\right) = \sum_{\sigma \in \Sigma^k} c\left(\sigma, \mathbf{x}\right) \cdot c\left(\sigma, \mathbf{x}'\right) \tag{2}$$

But, it has been understood by our previous studies [9], [10], that the spectrum kernel method alone does not show satisfactory classification performances. On the other hand, however, this method has the capability of increasing the total performance once combined with other methods.

3 Support Vector Machines and Committee SVM Array

3.1 Kernel Support Vector Machine

The support vector machine shows a high performance when an appropriate nonlinear transformation of source data $\phi_i(\mathbf{x})$ is used. Here, i stands for the index for different nonlinear transformations which will appear in committee SVMs. Given a data vector $\mathbf{x}$, each decision is made according to the margin.

$$f_i\left(\mathbf{x}\right) = \mathbf{w}_i^T \phi_i\left(\mathbf{x}\right) + b_i. \tag{3}$$

Here, $\mathbf{w}_i$ and b_i are learned values from training data. The nonlinear function $\phi_i(\mathbf{x})$ needs to satisfy the kernel property so that direct computations of the nonlinear transformation $\phi_i(\mathbf{x})$ can be avoided on the computation of the kernel

$$K_i(\mathbf{x}, \mathbf{x}') = \phi_i(\mathbf{x})^T \phi_i(\mathbf{x}'). \tag{4}$$

In later experiments, $K_i(\mathbf{x}, \mathbf{x}')$ will be radial basis functions with various dimensions and parameters, as well as the counting kernel of Equation (1).

3.2 Committee SVM Array

It has been observed well that the support vector machine shows better recognition performance than the neural network. This is because the SVM is obtained by the direct optimization of the performance measure. But, the SVM still has inevitable weak points:

(1) Multi-class decisions require practically infeasible computation. In fact, open SVM tools are mostly dichotomy machines. There are few examples for multi-class machines, however, the number of classes is very limited due to the complexity blow-up.

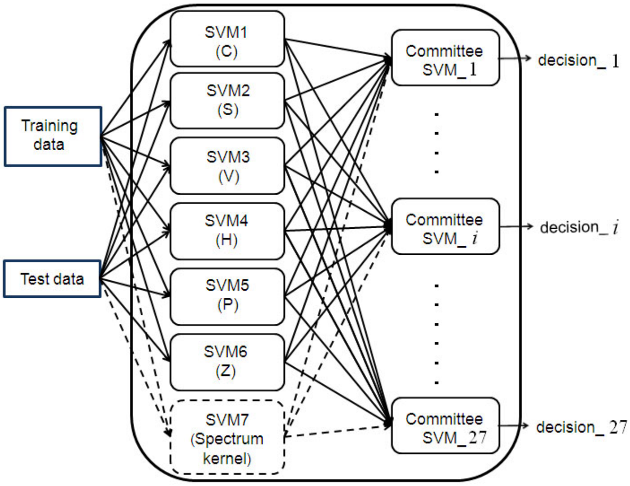


Fig. 3. Committee SVM

(2) If an input vector is a composition of very heterogeneous sub-vectors and has a high dimension, then the data space possesses many sparse sub-spaces. If implicit connections were created between such sub-spaces, the generalization performance would become low. This is a flaw common to large learning machines.

By considering the above (1), we adopt the strategy of "one-versus-others." On item (2), we design committee SVMs, i.e., multi-level SVMs. In this paper, they are two-level committee SVMs. Figure 3 illustrates the arrangement of such committee SVMs. First-level SVMs work on heterogeneous data explained in Section 2. At the second level, there are 27 SVMs prepared. Each SVM corresponds a specific feature vector or a spectrum kernel explained in Section 2. Input vectors to the second-level SVMs are the set of normalized margins or *soft decisions* from the first-level SVMs. The second level SVMs give decisions on 27 SCOP classes. Therefore, the committee SVMs of "one-versus-others" can be interpreted as a *committee SVM array.*

4 Experiments

4.1 Method of Performance Measurement

ROC Curve: There are three authoritative performance figures {specificity, sensitivity, precision}. They are defined as follows.

$$Specificity = \frac{TrueNegative}{TrueNegative + FalsePositive} \tag{5}$$

$$Sensitivity = \frac{TruePositive}{TruePositive + FalseNegative} \tag{6}$$

$$Precision = \frac{TruePositive + TrueNegative}{TruePositive + FalseNegative + FalsePositive + TrueNegative} \quad (7)$$

But, it is well-known that these figures are influenced by the ratio of positive and negative data. Therefore, the Receiver Operating Characteristic curve (ROC curve) is used as a more reliable performance measure. An ROC curve can be obtained by shifting a decision threshold and plot a pair (1-specificity, sensitivity) on an x-y plane.

Cross Validation: It is very likely that the content of positive and negative data is statistically biased by wrong sampling. Therefore, we draw n sets of positive and negative data. This enables to perform experiments n times. This is the n-fold cross validation. In our experiments, there are 694 data [5]. They are separated to 341 training data and 353 test data, which are generated by randomly halving each 27 SCOP classes. Thus, n=10 sets of test data and training data were generated. Finally, their average performance was computed as an ROC curve.

4.2 Experiments on Single One-Versus-Others SVM

We performed experiments on one-versus-others SVMs on 27 SCOP classes. This has the purpose of obtaining baseline performances for later experiments for the committee SVM.

Figure 4 illustrates ROC curves using one of {C, S, H, V, P, Z}. Each curve is the result of the average of 27 SCOP class decision. This illustration tells that the ordering of the feature effectiveness is C>S>H>P>V>Z.

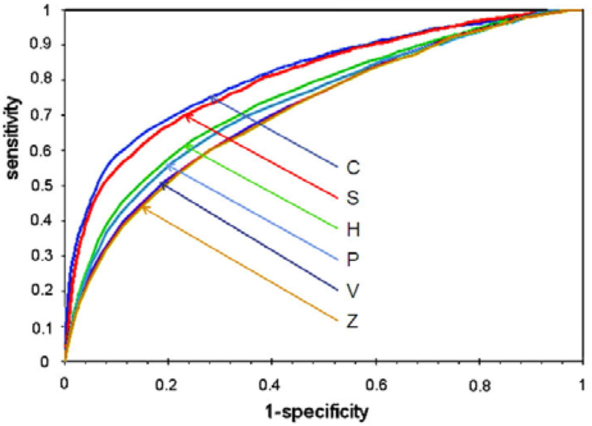


Fig. 4. ROCs of {C, S, H, V, P, Z}

4.3 Experiments Including CSHVPZ Super Vector

Figure 5 shows the ROC curve by the CSHVPS super vector which outperforms any of single SVMs of {C, S, H, V, P, Z}. Note that the ROC curve by the

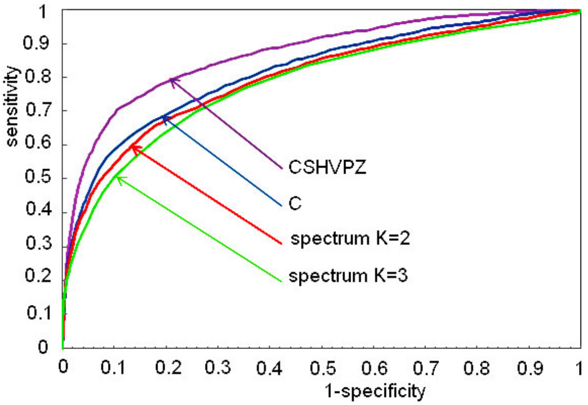


Fig. 5. ROCs of CSHVPZ super vector SVM, C-SVM, kernel spectrum SVMs of $k = 2,\ 3$

feature C is the same one in Figure 4. In Figure 5, the ROC curves by the spectrum kernel SVMs of $k = 2$ and $k = 3$ are also illustrated. The comparison of the super vector's ROC and the ROC curve for the feature C appeals the classification improvement by the super vector. The ROC curves by the spectrum kernels of $k = 2$ and $k = 3$ are illustrated in this figure, not in Figure 4, because the superposition of these curves would create a more confusing illustration of Figure 4.

4.4 Experiments on Committee SVM Array: Final One

Since the entities of {C, S, H, V, P, Z} and the spectrum kernel are very different, it becomes a good challenge to realize a total committee SVM array illustrated in Figure 3 if this scheme is within the conventional computing resource. The answer is yes. This is because each SVM is of one-versus-others type.

Figure 6 shows the following ROC curves which was obtained by using a conventional desktop computer of 2.00 GHz.

CSHVPZ : A single SVM using super vector data of CSHVPZ which is the ROC curve in Figure 5.

CM_CSHVPZ : A committee SVM by {C, S, H, V, P, Z}.

CM_CSHVPZK2 A committee SVM by {C, S, H, V, P, Z} and the spectrum kernel of $k = 2$.

CM_CSHVPZK3 A committee SVM by {C, S, H, V, P, Z} and the spectrum kernel of $k = 3$.

Before analyzing Figure 6 in detail, we give a very important notice here: *This graph is a partial enlargement of ROC curves.* This is because we need to pay attention to the most meaningful part of the ROC curve and the diffence in this part needs to be understood clearly. Figure 6 illustrates total types of

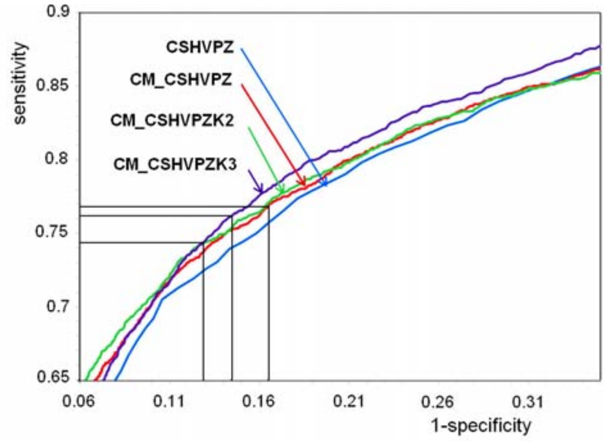


Fig. 6. ROCs of committee SVMs

resulting ROC curves on protein folding multi-classification. In the region where the specificity and the sensitivity are well-balanced, the committee SVM array for CM_CSHVPZK3 passes the point of (specificity, sensitivity)=(85.40%, 76.13%), CM_CSHVPZK2 goes through (87.2%, 74.4%), and the CM_CSHVPZ passes (83.3%, 76.67%). This shows a substantial improvement on the recognition of the folding classification.

5 Conclusion

The prediction of protein folding class from amino acid sequences was addressed. Since extracted features from one sequence are heterogeneous, a layered structure of support vector machines showed a substantial improvement on the classification performance over existing ones. The layered SVM, i.e., the committee SVM array, can avoid designing a large monolithic multi-class SVM which is practically infeasible to run on a conventional computer.

References

1. Mount, D.W.: Bioinformatics. Cold Spring Harbor Laboratory Press (2001)
2. Murzin, A.G., Brenner, S.E., Hubbard, T., Chothia, C.: SCOP: a structural classification of proteins database for the investigation of sequences and structures. J. Mol. Biol. 247, 536–540 (1995)
3. Dubchak, I., Muchunik, I., Holbrook, S.R., Kim, S.-H.: Prediction of protein folding class using global description of amino acid sequence. Proc. Natl. Acad. Sci. USA 92, 8700–8704 (1995)
4. Dubchak, I., Muchnik, I., Mayor, C., Dralyyuk, I., Kim, S.-H.: Recognition of a Protein Fold in the Context of the SCOP Classification. Proteins: Structure, Function, and Genetics 35, 401–407 (1999)

5. Ding, C.H.Q., Dubchak, I.: Multi-class protein fold recognition using support vector machines and neural networks. Bioinfo. 17, 349–358 (2001)
6. Leslie, C., Eskin, E., Noble, W.S.: The Spectrum kernel: A string kernel for SVM protein classification. Pacific Symposium on Biocomputing 7, 566–575 (2002)
7. Tabrez, M., Shamim, A., Anwaruddin, M., Nagarajaram, H.A.: Support vector machine-based classification of protein folds using the structural properties of amino acid residues and amino acid residue pairs. Bioinfo. 23, 3320–3327 (2007)
8. Lodhi, H., Saunders, C., Shawe-Taylor, J., Watkins, C.: Text classification using string kernels. J. of Machine Learning Research 2, 419–444 (2002)
9. Matsuyama, Y., Ishihara, Y., Ito, Y., Hotta, T., Kawasaki, K., Hasegawa, T., Takata, M.: Promoter recognition involving motif detection: Studies on E. coli and human genes. Intelligent Systems for Molecular Biology, Vienna, H06 (2007)
10. Matsuyama, Y., Kawasaki, K., Hotta, T., Mizutani, T.M., Ishida, A.: Eukaryotic transcription start site recognition involving non-promoter model. Intelligent Systems for Molecular Biology, Toronto, L05 (2008)

Implementation of the MLP Kernel

Cheng-Yuan Liou* and Wei-Chen Cheng

Department of Computer Science and Information Engineering
National Taiwan University
Republic of China
cyliou@csie.ntu.edu.tw

Abstract. This paper presents a MLP kernel. It maps all patterns in a class into a single point in the output layer space and maps different classes into different points. These widely separated class points can be used for further classifications. It is a layered feed-forward network. Each layer is trained using the class differences and trained independently layer after layer using a bottom-up construction. The class labels are not used in the training process. It can be used in separating multiple classes.

1 Introduction

The support vector machine (SVM) [1] employs the Mercer kernel to map patterns to the high dimensional space. Usually, the class information of the pattern is not used in the design of the kernel function. The outcome of the mapping relies on the choice and the setting of the kernel function.

This paper devises a method for the training of the MLP [7][8] and uses the trained MLP as the mapping kernel to split different classes. This trained MLP mapping kernel will be named 'SIR kernel' after the method in [7]. The SIR kernel can be applied to multiple classes. The design idea of the SIR kernel is in the following context.

Let the set of all patterns be $X = \{\mathbf{x}^p; p = 1, \ldots, P\}$. Each pattern $\mathbf{x}^p$ is a D-dimensional column vector. The label function, $C : \mathbb{R}^{n_0} \to \mathbb{N}$, maps each pattern, $\mathbf{x}^p$, to its class identity number (or class label), c_p. Let the set U_{c_i} contain all pattern pairs that belong to the same class c_i, $U_{c_i} = \{(\mathbf{x}^p, \mathbf{x}^q); C(\mathbf{x}^p) = C(\mathbf{x}^q) = c_i\}$. Let the set V_{c_i,c_j} contain all pattern pairs that belong to the different classes, $V_{c_i,c_j} = \{(\mathbf{x}^p, \mathbf{x}^q); C(\mathbf{x}^p) = c_i, C(\mathbf{x}^q) = c_j, c_i \neq c_j\}$.

Suppose there are L hidden layers in the network. Let the column vector $\mathbf{y}^{(p,m)}$ be the output vector of all neurons in the mth layer, $m \in \{1, 2, ..., L\}$, when the pattern $\mathbf{x}^p$ is fed to the input layer. $\mathbf{y}^{(p,m)}$ is the internal representation of $\mathbf{x}^p$ in the mth layer. We set $\mathbf{y}^{(p,0)} = \mathbf{x}^p$. Let n_m denote the total number of neurons in the mth layer. The collection of all internal representations of the mth layer is $\mathbf{Y}^m = \left\{\mathbf{y}^{(p,m)}, p = 1, \ldots, P\right\}$. The representations may be the same, $\mathbf{y}^{(p,m)} = \mathbf{y}^{(q,m)}$, for different patterns, $\mathbf{x}^p \neq \mathbf{x}^q$. This is a many-to-one mapping. Let $\|\mathbf{Y}^m\|$ be the total number of distinct representations in

* Corresponding author.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 378–385, 2009.

the set $\mathbf{Y}^m$. The internal representations of patterns, $\mathbf{Y}^m$, have been studied in the work [6]. All patterns have their representations in each layer. These representations are the output vectors of the layer for all input patterns. These representations are all binary codes when the hard limiting activation function is applied to all neurons. So, $\mathbf{y}^{(p,m)}$ is a binary code. The decision hyperplanes of all neurons in a layer divide its input layer space into nonoverlapped small decision areas and code these areas with binary codes. For the first hidden layer, each area has a polyhedral shape in the input pattern space. The inputs of the second layer are the binary internal codes of the outputs from the first hidden layer. Each of the codes, $\mathbf{y}^{(p,m)}$, represents the all patterns in a single decision area. According to the study in [6], the total number of representations will be much reduced, generally, in a layer that is far from the input layer. This means $\left\|\mathbf{Y}^L\right\| \ll .. \ll \left\|\mathbf{Y}^2\right\| \ll \left\|\mathbf{Y}^1\right\| \ll P$. This reduced number is very useful for the separation of classes. Ideally, this number can be reduced to the number of classes, $\left\|\mathbf{Y}^L\right\|$ =total number of classes=$\|C\|$. This makes the design possible for the SIR kernel.

The method in [5] devised a weight design for each layer. According to the design, the upper bound of the number of neurons in the mth layer required for solving a general-position two-class classification problem is $\left\lceil \frac{\left\|\mathbf{Y}^{m-1}\right\|}{n_{m-1}} \right\rceil \geq n_m$. For the number of neurons in the first hidden layer, n_1, the bound is $\left\lceil \frac{P}{D} \right\rceil \geq n_1$. With this weight design, the reduced number in the last layer L is guaranteed, $\left\|\mathbf{Y}^L\right\| = \|C\|$.

The work [6] also introduced a layered binary tree, named 'AIR' tree, that can trace the error neurons in a latent hidden layer that is far from the output layer and close to the input layer. The error neurons show that certain mixed patterns from both classes are represented in a same code (or included in a decision area). This means that a single code $\mathbf{y}^{(p,m)}$, $\mathbf{y}^{(p,m)} = \mathbf{y}^{(q,m)}$, represents different class patterns, $(\mathbf{x}^p, \mathbf{x}^q) \in V_{c_i,c_j}$. The joint nodes of the tree are the internal codes. According to this tree, any BP algorithm cannot correct such latent errors by adjusting its succeeding layers that near the output layer. The front layers must be trained correctly in order to send right signals to their succeeding layers. This suggests that one has to accomplish the MLP layer after layer using a bottom-up construction.

The study in the work [6] further identifies the mechanism of the front layers during the supervised BP training. It shows that categorization into different classes is the main mechanism for those front layers. This means that the identity of each class, the class label, is not used in the categorization. This suggests that the front layers can be successfully trained by using the discrimination differences between classes as the object function.

The SIR method in [7] provides such object function based on the differences between classes. The front layers can be trained layer after layer using this object function starting from the first hidden layer. Perfect categorization and production of right signals can be accomplished for each layer [7][8]. These front layers are served, suitably, as the SIR kernel. The SIR kernel will utilize the differences between classes to train the front layers. The kernel will not use the class label information in its training process.

The work [6] also identifies the mechanism of the rear layers that are near the output layer. It shows that labeling is the main mechanism. These front and rear mechanisms complete the supervised MLP. We will include a labeling sector that contains several layers after the SIR kernel. The object function for the labeling sector is the class labels.

We will use the differences between classes to train each front layer starting from the first hidden layer. A second hidden layer is added to the first hidden layer when the outputs of the first hidden layer cannot produce well isolated signals for each class. When a hidden layer can produce well isolated signals, it will be served as the last front layer, L, and as the output of the kernel. We expect that the number of reduced representations of the last front layer will be equal to the number of classes, $\left\|\mathbf{Y}^L\right\| = \|C\|$.

2 Method

Figure 1 shows the SIR kernel and the labeling sector. The SIR kernel consists of layered neurons.

For the pair patterns in the same class, $(\mathbf{x}^p, \mathbf{x}^q) \in U_{c_i}$, each layer is trained by using the energy function [8],

$$E^{att}\left(\mathbf{x}^p, \mathbf{x}^q\right) = \frac{1}{2}\left\|\mathbf{y}^{(p,m)} - \mathbf{y}^{(q,m)}\right\|^2, \tag{1}$$

to reduce the distance between their output vectors, $\left\|\mathbf{y}^{(p,m)} - \mathbf{y}^{(q,m)}\right\|$. For the pair, $(\mathbf{x}^p, \mathbf{x}^q) \in V_{c_i,c_j}$, each layer is trained by using the energy function,

$$E^{rep}\left(\mathbf{x}^p, \mathbf{x}^q\right) = \frac{-1}{2}\left\|\mathbf{y}^{(p,m)} - \mathbf{y}^{(q,m)}\right\|^2, \tag{2}$$

to increase the distance between their output vectors. The difference information between classes is implicitly used in the two energies. Note that the class labels are not used in these two object functions. The labels will be applied only in the labeling sector.

The network is constructed layer after layer, starting from $L = 1$. A new hidden layer is added, $L^{new}{=}L^{old} + 1$, whenever L^{old} layers cannot accomplish the isolation. All weights of the trained L^{old} layers are fixed during the training of the added layer, $m = L^{old} + 1$.

The weight matrix which connects the output of the $(m-1)$th layer and the input of the mth layer, is denoted by W_m. The W_1 connects the input layer and the first hidden layer. In this paper, 'W_m' is used for representing the weight matrix of the mth layer. Applying the gradient descent method to the added layer, the two energies can be reduced efficiently during training iterations. The successfully trained network is used as the SIR kernel to map the pattern, $\mathbf{x}^p$, to the output space, $\mathbf{y}^{(p,L)}$.

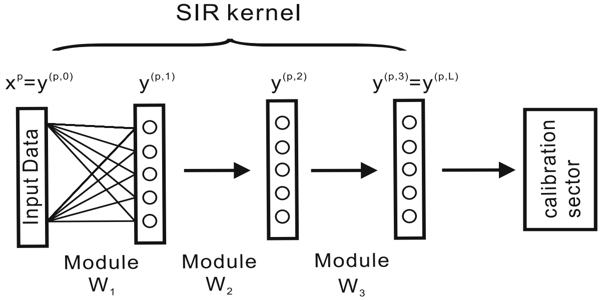


Fig. 1. The SIR kernel and labeling sector

Algorithm. Each time a new hidden layer, $L^{new} = L^{old} + 1$, is added, its weights are adjusted by the gradient descent method using the energies (1) and (2). The weights of all trained layers, L^{old}, are fixed. Suppose there are two classes, $\{c_1 = 1, c_2 = 2\}$. The training algorithm is in the followings:

1. For the added layer W_m, W_m from W_1 to W_L
2. For limited epochs
3. Pick two patterns in the same class, $\mathbf{x}^p$ and $\mathbf{x}^q$, which satisfy the following condition

$$(\mathbf{x}^p, \mathbf{x}^q) = \underset{\{(\mathbf{x}^i,\mathbf{x}^j)\in U_1 \ or \ (\mathbf{x}^i,\mathbf{x}^j)\in U_2\}}{\arg\max} \left\| \mathbf{y}^{(i,m)} - \mathbf{y}^{(j,m)} \right\|^2 . \tag{3}$$

Among the pair patterns in the same class, either in U_1 or in U_2, the two patterns $(\mathbf{x}^p, \mathbf{x}^q)$ have the longest distance in the output space of the mth layer.

4. Find the pair patterns, $\mathbf{x}^r$ and $\mathbf{x}^s$ in different classes, which satisfy

$$(\mathbf{x}^r, \mathbf{x}^s) = \underset{(\mathbf{x}^i,\mathbf{x}^j)\in V_{1,2}}{\arg\min} \left\| \mathbf{y}^{(i,m)} - \mathbf{y}^{(j,m)} \right\|^2 . \tag{4}$$

The pair patterns $(\mathbf{x}^r, \mathbf{x}^s)$ have the shortest distance in the output space of the mth layer.

5. Adjust the weight W_m by

$$\nabla W_m \leftarrow \eta^{att} \frac{\partial E^{att}(\mathbf{x}^p, \mathbf{x}^q)}{\partial W_m} + \eta^{rep} \frac{\partial E^{rep}(\mathbf{x}^r, \mathbf{x}^s)}{\partial W_m}$$
$$W_m \leftarrow W_m - \nabla W_m ,$$

where η^{att} and η^{rep} are learning rates.

3 Experimental Analysis

Two artificial datasets are used in the simulations. One is a two-class problem and the other is a three-class problem. Three real world datasets are also used in the simulations.

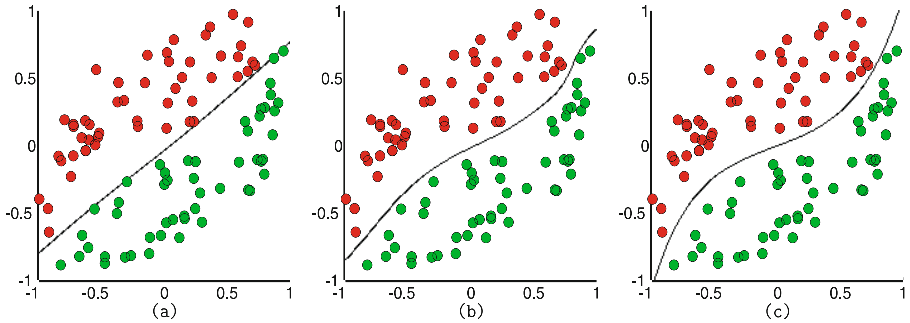


Fig. 2. (a) The training result of MLP. (b) The result of SIR kernel. (c) The result of SVM.

Two-Class Problem. Figure 2(b) shows the trained result for the two-class patterns, $c_i \in \{1, -1\}$, in the 2D plane, $n_0 = 2$. The border of the two-class patterns is $(x_1)^3 + 0.1x_1 = x_2$. Pattern points with the same color are in the same class. There are five neurons in each layer, $\{n_m = 5,\ m \in \{1, \ldots, L\}\}$. The kernel is trained layer after layer until it produces well isolated signals for each class. We set the isolation condition for inter-class representations as

$$\min_{(\mathbf{x}^p, \mathbf{x}^q) \in V_{1,2}} \left\| \mathbf{y}^{(p,L)} - \mathbf{y}^{(q,L)} \right\|^2 \approx 2^2 \times n_L, \tag{5}$$

and the condition for intra-class patterns as

$$\max_{\{(\mathbf{x}^p, \mathbf{x}^q) \in U_1 \text{or } (\mathbf{x}^p, \mathbf{x}^q) \in U_2\}} \left\| \mathbf{y}^{(p,L)} - \mathbf{y}^{(q,L)} \right\|^2 \approx 0. \tag{6}$$

The learning rates are $\eta^{att} = 0.01$ and $\eta^{rep} = 0.1$. The successful isolation is reached when $L = 2$. We set one neuron, $n_1^c = 1$, in the labeling sector as the output layer and use the class identities, $c_i \in \{1, -1\}$, to train this neuron. Figure 2(b) shows the trained result.

We also compare the result with those obtained by the MLP in Figure 2(a), and SVM in Figure 2(c). The MLP is a multilayer perceptron with two hidden layers, $n_1^{MLP} = n_2^{MLP} = 5$. This MLP is trained by the supervised BP. The polynomial kernel, $K(\mathbf{u}, \mathbf{v}) = \left(\mathbf{u}^T\mathbf{v} + 1\right)^3$, is used in SVM [2].

The boundary in Figure 2(b) is much more close to the intrinsic border than the result of the supervised MLP in Figure 2(a). The boundary learned by SVM in Figure 2(c) is also close to the intrinsic border. Using the Gaussian kernel, the SVM learned a similar boundary as that in Figure 2(a).

Multiple-Class Problem. The training patterns sampled from three classes separated by concentric circles, $c_i \in \{1, 2, 3\}$, are used in this simulation, see the right column in Figure 5. We train four SIR kernels with different number of neurons in each layer, $\{n_m = 5, n_m = 7, n_m = 9, n_m = 11\}$. Each layer is trained

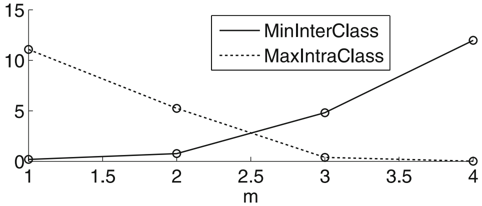


Fig. 3. Recorded isolation conditions for the case $n_m = 5$, MinInterClass (7) and MaxIntraClass (8), for each layer, $m = 1, 2, 3, 4$

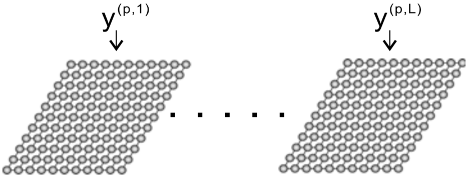


Fig. 4. The SOM used for visualization

with 1000 epochs. The isolation condition (6) is used in this simulation to stop the addition of a new layer. The learning rates are $\eta^{att} = 0.01$ and $\eta^{rep} = 0.1$. The values of the isolation conditions of each layer

$$MinInterClass(m) = \min_{(\mathbf{x}^p,\mathbf{x}^q)\in\{V_{1,2},V_{1,3},V_{2,3}\}} \left\| y^{(p,m)} - y^{(q,m)} \right\|^2 \tag{7}$$

and

$$MaxIntraClass(m) = \max_{(\mathbf{x}^p,\mathbf{x}^q)\in\{U_1,U_2,U_3\}} \left\| y^{(p,m)} - y^{(q,m)} \right\|^2 , \tag{8}$$

are recorded and plotted for the case $n_m = 5$ in Figure 3.

When the well isolation is reached, we set two layers in the labeling sector with $n_1^c = 2$ and $n_2^c = 3$ and use the class identities to train these two layers. In the layer $n_2^c = 3$, each neuron represents a single class.

We employ the SOM (Self-Organizing Map) [4] to visualize the output signals, $\mathbf{y}^{(p,m)}$, of each layer, to see the isolation of classes. The neurons of the SOM are placed on regular points, see Figure 4. The SOM consists of 10 × 10 neurons.

Figure 5 shows the SOM results for all layers. Each pixel denotes a SOM neuron. The pattern color is marked on its winner neuron. Figure 5 shows that well isolated signals are gradually accomplished in the last few layers. The output signals of the last layer have three concentrated points in the SOM.

Real Dataset. The iris dataset, Wisconsin breast cancer database, and Parkinson dataset will be used in the experiments. The iris dataset [3] contains 150 data items which belong to four classes. Each data is a four dimensional vector. The Wisconsin breast cancer database is a diagnostic dataset. Many useful attributes are used for the prediction of benign or malignant, a two-class problem. The study in [11] reported a 95.9% testing accuracy. The breast cancer

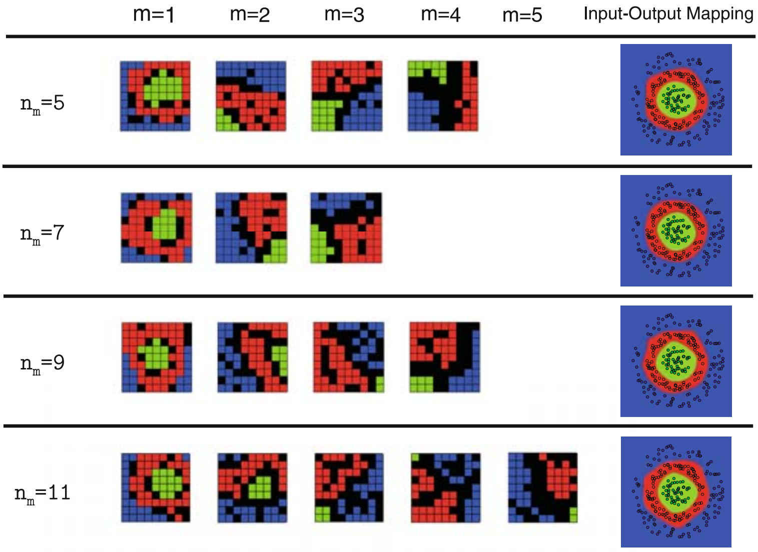


Fig. 5. The results of using the SOM to visualize the isolation of the output vectors of each layer. The images on the right column display the mapping relation between input patterns and output space.

Table 1. Parameters

	SIR	SVM		MLP	
	$(n_m, L_{\max})$	C	gamma	n_1^{MLP}	n_2^{MLP}
iris	$(11, 5)$	50	0.05	20	10
Wisconsin Breast Cancer	$(30, 7)$	50	0.05	30	10
Parkinson	$(20, 5)$	50	0.05	30	10

Table 2. The accuracy on real dataset

	Training Accuracy			Testing Accuracy		
	SIR	MLP	SVM	SIR	MLP	SVM
iris	100.00%	99.67%	97.50%	97.33%	94.66%	96.00%
Wisconsin Breast Cancer	100.00%	98.89%	97.53%	96.00%	95.57%	96.42%
Parkinson	100.00%	98.33%	99.87%	91.28%	88.20%	92.82%

dataset has 16 missing values. These missing values are set to zero. The Parkinson dataset [9] contains the biomedical voice measurement by healthy people and parkinson patients.

Three machine learning techniques, SIR kernel, MLP and SVM, are compared using the 5-fold cross-validation. The dataset is randomly split into five partitions, four of them are used in the training process and the rest one is used in the testing process. The result is the average of the 5-fold cross-validation. The labeling sector for the iris set is $n_1^c = 5$ and $n_2^c = 3$. The sector for the cancer dataset is $n_1^c = 5$ and $n_2^c = 1$. The sector for the Parkinson dataset is $n_1^c = 5$ and $n_2^c = 1$. The largest value L among the five trials is listed in the Table 1 under the column marked '$L_{\max}$'. The parameters of SVM are the cost C for the error tolerance and the gamma in the Gaussian kernel. The MLP has two hidden layers. All parameters are listed in Table 1. The values of the input data are normalized to the range $[-1, 1]$.

Table 2 shows that the SIR kernel is competitive and practicable in real world applications. The column 'SIR' contains the results of SIR kernel.

Acknowledgement. This work is supported by National Science Council.

References

1. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A Training Algorithm for Optimal Margin Classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, New York, NY, USA, pp. 144–152 (1992)
2. Chang, C.-C., Lin, C.-J.: LIBSVM: a Library for Support Vector Machines (2001), http://www.csie.ntu.edu.tw/~cjlin/libsvm
3. Fisher, R.A.: The Use of Multiple Measurements in Taxonomic Problems. Annual Eugenics, Part II 7, 179–188 (1936)
4. Kohonen, T.: Self-Organized Formation of Topologically Correct Feature Maps. Biological Cybernetics 43, 59–69 (1982)
5. Liou, C.-Y., Yu, W.-J.: Initializing the Weights in Multilayer Network with Quadratic Sigmoid Function. In: Proceedings of International Conference on Neural Information Processing, pp. 1387–1392 (1994)
6. Liou, C.-Y., Yu, W.-J.: Ambiguous Binary Representation in Multilayer Neural Network. In: Proceedings of ICNN, vol. 1, pp. 379–384 (1995)
7. Liou, C.-Y., Chen, H.T., Huang, J.C.: Separation of Internal Representations of the Hidden Layer. In: Proceedings of the International Computer Symposium, pp. 26–34 (2000)
8. Liou, C.-Y., Cheng, W.-C.: Resolving Hidden Representations. In: Ishikawa, M., Doya, K., Miyamoto, H., Yamakawa, T. (eds.) ICONIP 2007, Part II. LNCS, vol. 4985, pp. 254–263. Springer, Heidelberg (2008)
9. Little, M.A., McSharry, P.E., Roberts, S.J., Costello, D.A.E., Moroz, I.M.: Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection. BioMedical Engineering OnLine 6, 23 (2007)
10. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning Internal Representations by Error Propagation. In: Rumelhart, D.E., McClelland, J.L. (eds.) Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, pp. 318–362. MIT Press, Cambridge (1986)
11. Wolberg, W.H., Mangasarian, O.L.: Multisurface Method of Pattern Separation for Medical Diagnosis Applied to Breast Cytology. In: Proceedings of the National Academy of Sciences, vol. 87, pp. 9193–9196 (1990)

Fuzzy Rules Extraction from Support Vector Machines for Multi-class Classification with Feature Selection

Adriana da Costa F. Chaves, Marley Vellasco, and Ricardo Tanscheit

Electrical Engineering Department,
Pontifical Catholic University of Rio de Janeiro
nacha@ele.puc-rio.br, marley@ele.puc-rio.br,
ricardo@ele.puc-rio.br

Abstract. - Although Support Vector Machines (SVMs) have been successfully applied to many problems, they are considered "black box models". Some methods have been developed to reduce this limitation, among them the FREx_SVM, which extracts fuzzy rules from trained SVMs for multi-class problems. This work deals with an extension to the FREx_SVM method, including a wrapper feature subset selection algorithm for SVMs. The method was evaluated in four benchmark databases. Results show that the proposed extension improves the original FREX_SVM, providing better rule coverage and a lower number of rules, which is a considerable gain in terms of interpretability.

1 Introduction

Support vector machines (SVMs) are based on statistical learning theory [1, 2, 3] and have been applied with excellent generalization performance to a variety of applications in classification and regression [4, 5, 6].

Despite their excellent performance, SVMs, as well as artificial neural networks, are "black box models", i.e., models that do not explain clearly what leads to a given result. Algorithms whose purpose is to extract useful knowledge from a trained SVM have already been proposed, among them RulExtSVM [7] and SVM+Prototypes [8]. The algorithm RulExtSVM extracts IF-THEN rules with intervals, defined by hyper-rectangular forms, in the rules' antecedents. The SVM+Prototypes method calculates ellipsoids (called prototypes) based on the obtained support vectors of each class. These ellipsoids are also used in the rules' antecedents.

It must be stressed that the rules extracted from both methods generate, in their antecedents, intervals or functions. This fact reduces the interpretability of the generated rules and jeopardizes the capacity of knowledge extraction [9]. To increase the linguistic interpretability of the generated rules, the FREx_SVM method has been developed [9]. It extracts fuzzy rules from trained support vector machines and is applicable to multi-class classification problems. The basic idea is that, by employing fuzzy sets in the rules' antecedents, the resulting rules will be more flexible and interpretable. The FREx_SVM method has already been successfully applied to several databases [9]. However, when the number of input attributes increases, the interpretability of the model decreases, since the generated rules have many variables in the antecedents. Therefore, this article proposes an extension to the original FREx_SVM by including a

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 386–393, 2009.

wrapper feature selection algorithm [10, 11, 12] to select the most relevant input features. Two feature selection algorithms, proposed in [10] and [12], were evaluated with FREx_SVM; however, preliminary results indicated that the former algorithm ([10]) provided better results with the original FREx_SVM. Therefore, the case studies carried out in this work were obtained with the feature selection algorithm presented in [10].

This paper is divided into five additional sections. Section 2 briefly describes the theory of support vector machines. The feature selection methods are presented in section 3. Section 4 briefly presents the method for extracting fuzzy rules from trained SVMs. First the method is presented for binary classification problems; then the fuzzy rule extraction model is extended to multi-classification applications. Section 5 presents the case studies, describing the benchmark databases employed and the performance obtained with the fuzzy rule extraction method. Finally, discussion and conclusion are presented in Section 6.

2 Support Vector Machines

Consider a training set $\{(xi, y_i)\}$, $i \in \{1, \ldots, N\}$, where $x_i \in \mathbb{R}^n$, $y_i \in \{-1, 1\}$, and N is the number of patterns. SVM solves a quadratic programming optimization problem:

$$\text{maximize} \quad \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{1}$$

$$\text{subject to} \quad 0 \le \alpha_i \le \mathrm{C}, \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0 . \tag{2}$$

The function $K(x_i, x_j) = \Phi(x_i).\Phi(x_j)$ is a kernel function, where $\Phi(x_i)$ represents the mapping of input vector x_i into a higher dimensional space (called “feature space”). C is the regularization constant, a positive training parameter which establishes a trade off between the model complexity and the training error, and α_i are Lagrange coefficients. In the solution of the previous problem, $\alpha_i = 0$ except for the support vectors, which are represented by s_i in order to distinguish them from other data samples. Methods for solving the above problem are described in [1, 2].

A data point x is classified according to the sign of the decision function:

$$f(\mathrm{x}) = \sum_{i=1}^{Ns} \alpha_i y_i K(s_i, x) + b \ , \tag{3}$$

where b is the bias and Ns is the number of support vectors.

SVMs were originally defined for binary classification. There are basically two approaches to extend SVM for classification in k classes (k > 2): one that reduces the k-classes problem to a set of binary problems; and one that involves a generalization of the binary SVM [13].

A well known method based on the first approach is "one-against-all" [13]. This method is the most simple and most used in the multi-class classification with SVM, so it was chosen to be employed in this paper. This method builds k binary SVMs, each of them dedicated to separating each class from the others. The outputs of all SVMs are then combined to generate the final classification in k classes. The most common method for combining the k SVMs outputs, which will be used in the case studies of this work, is to assign the input vector to the class that provides the largest value of the decision function.

3 Feature Subset Selection Algorithms

The algorithm for feature selection presented in [10] was designed specifically for support vector machines and employs the representation of the decision function given by equation 3.

The main idea is to extract some measure from a trained SVM and use this information to select the most relevant features in the training process. Then a new training phase is carried out with the input data containing the selected features.

In order to find the most relevant features, a measure of the importance of each feature is needed. Consider the linear case where w is the normal vector of the separating hyperplane and $\{e_{\text{i}}\}$ is a basis for the input space. The importance of a feature X_l is defined by the amount of colinearity of e_l with w. In the nonlinear case, this measure consists of the decision function f given by equation 3. However, the influence of a feature depends on the other features. Therefore, for a given point x in the input space, the importance d of X_l is defined as the partial derivative of the decision function f in relation to X_i:

$$d_{l,\vec{x}} = \left(\left\langle \nabla f(\vec{x}), \vec{e}_l \right\rangle\right)^2 \tag{4}$$

To evaluate the former equation, some points from the input space must be chosen, since it is impossible to evaluate the equation in all input points (there are infinite points). As the decision function depends only on the support vectors, these points will be selected.

A good measure of the influence d of a feature X_l is the mean of its influence computed in the support vectors:

$$d_l = \frac{1}{|sv|} \sum_{\vec{x}_i \in sv} \frac{d_{l,\vec{x}_i}}{\sum_l d_{l,\vec{x}_i}} \tag{5}$$

where $|sv|$ is the number of support vectors.

Once the importance d of each feature is calculated, the features are ranked. Then several SVMs are trained, the first with the best ranked feature, the second with the two best ranked features and so on. The training stops when the accuracy of a SVM trained on a feature set stops increasing.

In [12], authors proposed a prediction risk based feature selection method, which evaluates a feature by computing the change of training error when this feature is replaced by its mean value. The feature corresponding to the least change will be removed, since its change causes the least error indicating it is the least important one.

4 Fuzzy Rule Extraction Methodology

This paper concisely presents the method for extracting fuzzy rules from trained SVMs, called FREx_SVM. For further information, see [9]. The FREx_SVM method is divided in 3 steps: projection of support vectors, definition of fuzzy sets, and fuzzy rule extraction. For the sake of simplicity, the FREx_SVM method is first described for a binary SVM. Then, in Section 4.2, it is extended to a multi-class SVM.

4.1 FREx_SVM for Binary SVM

In the first step of the algorithm, the support vectors obtained by the SVM are projected on the coordinate axes. There are as many projections as the input space dimension (number of input attributes of the data base under analysis).

The next step consists of defining a number of overlapping fuzzy sets to each input variable. Each fuzzy set is labeled and assigned to a triangular membership function, usually with equal domain. Suppose that each attribute of an n-dimensional input space has been divided into m fuzzy sets. In this case, the fuzzy set C_{ij} is the j^{th} set defined for the i^{th} coordinate, where $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$.

Each support vector projection obtained in the previous step (one projection for each input variable) is then evaluated in all m membership functions, selecting the fuzzy set that provides the maximum membership degree.

In the final step, each support vector generates a fuzzy rule, as described below. For each support vector x, let C_{ij^i} be the fuzzy set with higher membership degree for each x_i coordinate, $i = 1 \ldots n$ and $j^i = 1 \ldots m$. The rule generated for support vector x is:

IF x_1 is C_{1j^1} and … and x_n is C_{nj^n} THEN $x = (x_1,\ldots,x_\text{n})$ belongs to the class defined by the support vector x.

In order to evaluate each rule, two metrics - *fuzzy accuracy* and *fuzzy coverage* - were defined to assess, respectively, how accurate the rule describes the data base and how many patterns are affected by the rule. Details of these proposed metrics are presented in [9].

4.2 FREx_SVM for Multi-class SVM

Since the proposed method is based on the support vectors obtained after training all SVMs, the binary approach presented in the previous section can be easily extended to multi-class problems.

As mentioned, the one-against-all method gives origin to k SVMs. If SVM$_i$ is the one that separates class i from all others, only support vectors from class i are considered in the generation of rules.

5 Case Studies

The goal of the case studies is to evaluate the impact of a good feature selection in the performance of FREx_SVM and, most importantly, its influence on the quality of the generated rules. Therefore, FREx_SVM is evaluated with and without the feature selection presented in [10]. Four benchmark databases, obtained from http://www.ics.uci.edu/~mlearn/MLRepository.html, have been used: Bupa Liver Disorders, Wisconsin Breast Cancer, Pima Indians Diabetes and Wine.

All SVMs were trained with two different kernel functions: linear and RBF (Gaussian functions). Three values for the regularization constant (parameter C, eq. 2) and four values for the Gaussian standard deviation were used to train each SVM: C = 0.1, 1 and 10 and σ^2 = 1, 5, 10 and 50. In the multi-class benchmark - Wine, the one-against-all method is employed for classification.

In order to evaluate the performance of FREx_SVM, two configurations of fuzzy sets (3 and 5 fuzzy sets), for each coordinate, were considered.

In all experiments the databases were divided into two equal size disjoint sets. These two sets were interchanged for training and testing, in two different experiments. The results presented in Table 1 and Table 2 are the average results in the test sets of these two experiments, with the best parameter configuration among all SVMs trained. The following performance metrics were used for comparing different configurations: the percentage of test examples that are covered by the rules (Cov), the classification percentage error of the generated rules (Err) and the number of generated rules (NR) for 3 (Table 1) and 5 fuzzy sets (Table 2). The best average results for the SVM in the test sets of the two experiments for each database are presented in Table 3 with the same metrics.

5.1 Bupa Liver Disorders

This database consists of medical information related to liver disorders with six numeric input attributes and one binary output.

In Table 1, without feature selection, the best Coverage (95.94%) was obtained with 28 rules. With feature selection, the number of features is defined as four. The best Coverage (98.27%) was obtained with 3 fuzzy sets, with only 12.5 rules in average, that is, FREx_SVM combined with the feature selection method improves by 02.33% the FREx_SVM Coverage, with less than half of the original number of rules. Besides, the error rate decreases by 05.23%.

5.2 Wisconsin Breast Cancer

This database, like Bupa Liver Disorders, consists of medical information related to breast cancer. There are nine numeric input attributes and one binary output.

The best Coverage (77.90%) without feature selection was obtained with 3 fuzzy sets, resulting in 131.5 rules in average. With feature selection, the number of features is set as six. The best Coverage (91.22%) was obtained for 3 fuzzy sets, with only 92.5 rules, that is, FREx_SVM combined with the feature selection method improves by 13.32% the FREx_SVM Coverage with fewer rules. The error rate increases, but only by 01.47%, which is less that 10% of the improvement in the Coverage.

5.3 Pima Indians Diabetes

This database is also related to medical information: analysis of diabetes. There are eight numeric input attributes and one binary output.

Without feature selection, the best Coverage (99.09%) was obtained for 3 fuzzy sets, with 118 rules. With feature selection, the number of features is defined as four. The best Coverage (99.87%) was obtained for 3 fuzzy sets, with only 30 rules, that is, FREx_SVM with the feature selection method improves only by 00.78% the FREx_SVM Coverage, but with almost one fourth of the original number of rules. Besides, the error rate decreases by 02.60%.

5.4 Wine

The last database tested is also a well known benchmark in pattern recognition literature. This database relates to three types of wine produced in a specific region of Italy. There are 13 numeric input attributes and, as usual, one classification output.

As can be seen from Table 1, without feature selection the best result attained the coverage of 92.13%, with 84 rules.

With feature selection, the number of features is defined as six. The best Coverage (97.75%) was obtained for 3 fuzzy sets, with only 44 rules, that is, FREx_SVM combined with the feature selection method improves 05.62% the FREx_SVM Coverage with almost half of the original number of rules. The error rate increases, but only 01.12%.

It is worth mentioning that for the 5-fuzzy set case, shown in Table 2, FREx_SVM combined with the feature selection method improves 44.94% the FREx_SVM

Table 1. Best Performance of FREx_SVM for 3 fuzzy sets

	Bupa Liver Disorders		**Wisconsin Breast Cancer**		**Pima Indians Diabetes**		**Wine**	
Features	6	4	9	6	8	4	13	6
Kernel	RBF $\sigma^2 = 50$ C = 0.1	RBF $\sigma^2 = 1$ C =10	RBF $\sigma^2 = 1$ C = 0.1	RBF $\sigma^2 = 50$ C = 0.1	RBF $\sigma^2 = 1$ C =0.1	RBF $\sigma^2 = 1$ C =1	RBF $\sigma^2 = 10$ C = 0.1	RBF $\sigma^2 = 10$ C = 0.1
Cov (%)	95.94	98.27	77.90	91.22	99.09	99.87	92.13	97.75
Err (%)	47.25	42.02	02.49	03.96	29.17	26.57	07.87	08.99
NR	28	12.5	131.5	92.5	118	30	84	44

Table 2. Best Performance of FREx_SVM for 5 fuzzy sets

	Bupa Liver Disorders		**Wisconsin Breast Cancer**		**Pima Indians Diabetes**		**Wine**	
Features	6	4	9	6	8	4	13	6
Kernel	RBF $\sigma^2 = 10$ C = 0.1	RBF $\sigma^2 = 50$ C =10	RBF $\sigma^2 = 1$ C = 0.1	RBF $\sigma^2 = 50$ C = 0.1	RBF $\sigma^2 = 1$ C =0.1	RBF $\sigma^2 = 1$ C =1	RBF $\sigma^2 = 5$ C = 0.1	RBF $\sigma^2 = 10$ C = 0.1
Cov (%)	85.51	94.50	63.98	72.62	83.46	98.83	51.69	96.63
Err (%)	40.87	44.04	00.73	01.32	35.15	35.55	00.00	05.62
NR	74	30	157	135.5	254	108	86	66

Table 3. Best Performance of SVM

	Bupa Liver Disorders		**Wisconsin Breast Cancer**		**Pima Indians Diabetes**		**Wine**	
Features	6	4	9	6	8	4	13	6
Kernel	Linear, C = 0.1	RBF $\sigma^2 = 1$ C =10	Linear C = 10	RBF $\sigma^2 = 10$ C = 1	RBF $\sigma^2 = 10$ C =1	RBF $\sigma^2 = 10$ C =1	RBF $\sigma^2 = 10$ C = 10	RBF $\sigma^2 = 1$ C = 10
Cov (%)	100.00	100.00	100.00	100.00	100.00	100.00	100.00	97.19
Err (%)	38.55	35.93	01.27	02.78	22.53	23.05	01.12	01.68

Coverage. The error rate increases 05.62%, but the gain in the Coverage is outstanding, which greatly improves the interpretability of the relation between the input variables and the associated output classification.

6 Conclusions

This paper presented an extension of the FREx_SVM method, which extracts fuzzy rules from a trained SVM, with a wrapper feature subset selection algorithm for SVM.

The main advantage of the FREx_SVM method is that the generated rules have fuzzy sets in their antecedents, which increases the linguistic interpretability. Additionally, the method can be applied to multi-class problems.

It must be stressed that the main goal of FREx_SVM is to extract interpretable knowledge from a trained SVM. Therefore, although the percentage errors provided by FREx_SVM in the four benchmark datasets evaluated are larger than the values provided by the SVMs, they are not really relevant in terms of how well the extracted rules help understanding the relation between the input vector and the output classification.

The extension of FREx_SVM, including a wrapper feature selection, increased the interpretability of the generated rules. The obtained results also provided a good improvement in rules Coverage (the percentage of test examples that are covered by the rules) and a remarkable reduction in the number of generated rules. Thus, the final model contains fewer rules, with fewer attributes in their antecedents, helping users´ understanding.

As for future work, the proposed algorithm shall include adaptive fuzzy sets. This shall improve even further accuracy and coverage of fuzzy rules, and possibly reduce the final number of extracted rules.

Acknowledgements

This research has been funded by the Rio de Janeiro Research Foundation (FAPERJ) under process number E-26/170.878/2007.

References

1. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and other kernel - based learning methods. Cambridge University Press, Cambridge (2000)
2. Haykin, S.: Neural Networks - A Comprehensive Foundation. Macmillan College Publishing Company, Basingstoke (1999)
3. Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons, Chichester (1998)
4. Vapnik, V.N., Golowich, S.E., Smola, A.: Support vector method for function approximation, regression estimation, and signal processing. In: Advances in Neural Information Processing System, vol. 9, pp. 281–287. Morgan Kaufmann, San Francisco (1997)

5. Drucker, H.D., Wu, D.H., Vapnik, V.N.: Support vector machines for spam categorization. IEEE Trans. on Neural Networks 10(5), 1048–1054 (1999)
6. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: Proceedings of the European Conference on Machine Learning, pp. 137–142. Springer, Heidelberg (1998)
7. Fu, X., Ong, C.J., Keerthi, S., Hung, G.G., Goh, L.: Extracting the Knowledge Embedded in Support Vector Machines. In: International Joint Conference on Neural Networks (IJCNN 2004), CDROM, Budapest, July 25-29 (2004)
8. Nunez, H., Angulo, C., Catalá, A.: Rule Extraction From support vectors machines. In: European Symposium on Artificial Neural Networks (ESANN), pp. 107–112 (2002)
9. Chaves, A., Vellasco, M.M.B.R., Tanscheit, R.: Fuzzy Rules Extraction from Support Vector Machines for Multi-class Classification. In: Analysis and Design of Intelligent Systems Using Soft Computing Techniques, ch. II. Advances in Soft Computing, vol. 41, pp. 99–108. Springer, Heidelberg (2007)
10. Heiler, M., Cremers, D., Schnörr, C.: Efficient Feature Subset Selection for Support Vector Machines. Technical Report 21/2001, Computer Sciences Series, University of Mannheim, Germany (2001)
11. John, G., Kohavi, R., Pfleger, K.: Irrelevant Features and the Subset Selection Problem. In: Proc. Of the 11th Int. Conf. on Machine Learning, pp. 121–129 (1994)
12. Moody, J., Utans, J.: Principled architecture selection for neural networks: Application to corporate bond rating prediction. In: Advances in Neural Information Processing Systems, vol. 4, pp. 683–690. Morgan Kaufmann Publishers, San Francisco (1992)
13. Hsu, C.-W., Lin, C.-J.: A Comparison on Methods for Multi-class Support Vector Machines. IEEE Transaction on Neural Networks 13(2), 415–425 (2002)

An SVM Based Approach to Cross-Language Adaptation for Indian Languages

A. Vijaya Rama Raju and C. Chandra Sekhar

Department of Computer Science and Engineering,
Indian Institute of Technology Madras, India
{ramaraju,chandra}@cse.iitm.ac.in

Abstract. In this paper we present an evaluation of different approaches to cross-language adaptation for Indian languages. We also propose a method for cross-language adaptation of the SVM (support vector machine) based system. The proposed method gives approximately the same performance as pooling, with a reduction in the training time. The adaptation methods such as Bootstrap, MAP (Maximum A Posterior) and MLLR (Maximum Likelihood Linear Regression) have been used for cross-language adaptation in the HMM (hidden Markov model) based systems. We present a comparison of these adaptation techniques for three Indian languages, Tamil, Telugu and Hindi. The results show that the SVM based methods perform better than the HMM based methods when the 2-best and 5-best performance is considered.

1 Introduction

Automatic speech recognition is essential for many spoken language applications, such as spoken document retrieval, speech summarization and browsing of meeting records. However, developing these systems requires large amounts of training data. Collection and annotation of large amount of speech data is very expensive in terms of labour and cost. Sometimes we may need to develop a speech recognition system for a language that has small amount of data. Building a good speech recognition system using a small amount of data is not possible. Cross-language adaptation is useful in such cases. In cross-language adaptation technique, a well trained speech recognition system of one or more languages will be adapted using a small amount of target language data, to build a speech recognition system for the target language. The adapted system will be used to recognize the speech of target language.

In order to adapt the speech recognition system of a source language to a target language, it is necessary to map the phonetic units of the target language to those of the source language. This can be done by using the phonetic knowledge like in IPA mapping [1], or a data-driven method or a combination of both. In [2], cross-lingual recognition was used for recognizing the speech of a language that was not used in training. The MLLR (Maximum Likelihood Linear Regression) technique, which was originally proposed for speaker adaptation [3], has been used for non-native speech adaptation and was compared with the MAP

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 394–401, 2009.

(Maximum A Posterior) adaptation technique in [4]. It is shown that the MAP technique performs better than the MLLR technique when large amount of data is available for adaptation. In [5], Schultz and Waibel compared cross-lingual recognition (without adaptation data), MLLR and bootstrap methods using different amounts of adaptation data. It was shown that the word error rate decreases as the amount of adaptation data increases. Zhao and Shaughnessy [6] presented a comparison of the performance of MLLR and native speech training for different amounts of adaptation data and found that the native speech training outperforms the MLLR adaptation, with training data of 12 minutes or more.

There is a lack of work on comparison of different adaptation methods with native speech training for Indian languages. This paper presents a comparison of native speech training and different adaptation methods for recognition of subword units of speech in Indian languages. We have built acoustic models for the frequently occurring consonant-vowel (CV) units. The support vector machines are well known for their ability to train with less data. But, there is a lack of work on cross-language adaptation of the SVM based system. This paper studies the effectiveness of the SVMs for cross-language adaptation using pooling. We also present a variation of the pooling method for cross-language adaptation of an SVM based system. In the proposed method, the support vectors of the source language system are pooled instead of the entire data of source language. This results in a reduction in the time required for adaptation by a factor of 4.

2 Cross-Language Adaptation Methods

In cross-language adaptation, the model built for a language (source language) is adapted to recognize the speech in a new language (target language) [2]. Figure 1 shows the block diagram of a cross-language adaptation system.

Some of the methods commonly used for cross-language adaptation are as follows:

1. Bootstrap adaptation
2. Transformation based adaptation
3. Bayesian adaptation

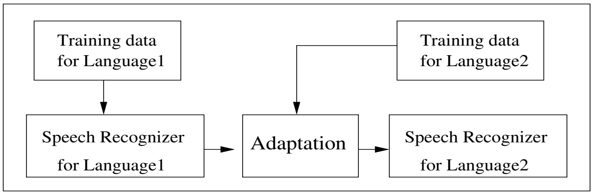

Training data for Language1
Training data for Language2
Speech Recognizer for Language1
Adaptation
Speech Recognizer for Language2

Fig. 1. Cross-language adaptation system

2.1 Bootstrap Adaptation

In this method, the well trained acoustic models of the source language are taken as seed models and are then retrained using the target language data. This method is suitable when large amount of target language data is available. In [7], the cross-language seed models performed better than the flat starts or random models.

2.2 Transformation Based Adaptation

If only a small amount of data is available, then the transformation based adaptation is useful. The Maximum Likelihood Linear Regression (MLLR) [3] is the most widely used method in this category. The MLLR computes a set of transformations to reduce the mismatch between the model set and the adaptation data. It computes a set of linear transformations for the mean and covariance parameters of Gaussian mixtures of an hidden Markov model (HMM) based system.

2.3 Bayesian Adaptation

In Bayesian adaptation, we use the prior knowledge about the model parameter distribution to make a good use of the limited adaptation data. The most popular implementation of Bayesian adaptation for speech recognition is to use the maximum a posteriori (MAP) estimator [8]. A combination of MLLR and MAP was used in [8] where the MLLR transformed models were used to seed the prior distributions for MAP estimation. This combination has given an improved performance for cross-language adaptation (English to Afrikaans) and also for cross-database adaptation (SUN speech English to TIMIT English).

3 Proposed Method for SVM Based System Adaptation

The above mentioned methods are applicable only to HMM based systems as they involve the updation of mean and covariance parameters of the models using adaptation data. As SVM based systems are discriminative models, they do not have any such kind of parameters. So, we need to use different methods for adaptation of SVM based systems. The proposed method for adaptation is a variation of the pooling method. In *pooling*, we combine the adaptation data with the source language data. This combined data is used to build an SVM model for each class using the one-against-the-rest strategy to multiclass classification. This method takes long time for adaptation. The source language speech recognition system was built using the one-against-the-rest strategy to multiclass classification. For the model of a particular class, the positive support vectors represent the examples of that class useful in discriminating the class from all the other classes. The negative support vectors are the examples of all the other classes useful for discriminating that class from the rest. The multiclass classification system using the one-against-the-rest SVM for a synthetic data set is shown in Fig.2. It shows

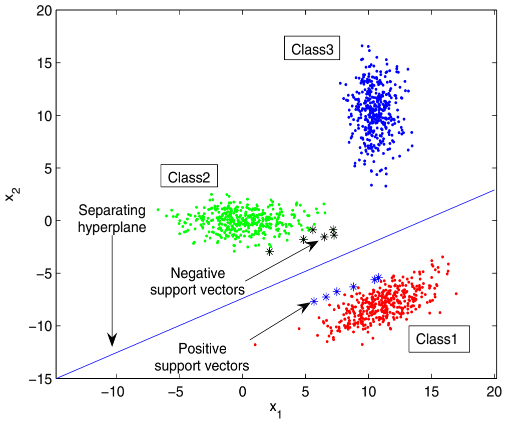


Fig. 2. Multiclass classification using support vector machines for synthetic data

the hyperplane separating the class1 from the remaining two classes and support vectors corresponding to that classifier. It can be observed from the Fig.2 that the number of support vectors is small compared to the number of total examples. In the proposed method for adaptation, we combine the support vectors of the SVM models for the source language with the adaptation data. The support vectors are the only examples useful for discriminating the classes. So, they form the sufficient examples to pool with the adaptation data instead of the entire data of source language. And, the number of support vectors is small compared to the number of total examples. Therefore, the models are expected to be adapted in less time. The steps involved in the proposed method for adaptation are as follows:

1. Train the SVM system with multiclass classification for the CV units of the target language using the adaptation data.
2. Identify the common CV units of the source and target languages. For each common CV unit, follow the steps 3 to 5.
3. Combine the positive support vectors of the model of the unit in the source language with the adaptation data of the corresponding unit.
4. Combine the negative support vectors of the model of the unit in the source language with the negative support vectors of the model for the corresponding unit in the target language.
5. Use these combined sets of examples to build an SVM for the CV unit to obtain an adapted model for the unit.

The first step ensures that the models for CV classes of the target language that are not present in the source language are trained against the other CV classes in the target language.

4 Speech Database

The database used in our studies was the speech recordings of the Doordarshan (TV channel) news bulletins in three Indian languages Tamil, Telugu and

Table 1. Description of the news database

Description	Language		
	Tamil	Telugu	Hindi
Number of bulletins	33	20	19
News readers (Male:Female)	(10:23)	(11:9)	(6:13)
#Bulletins used for source language	27	16	16
#Bulletins used for testing	6	4	3
Number of CV units	110	134	98

Hindi. A description of the database is given in Table 1. The data was collected from news bulletins, each of 10-15 minutes duration. Each bulletin was recorded by a single speaker. The syllables are segmented and labeled manually. These units have varying frequencies of occurrences in the database. In this work, the speech recognition systems were built for recognizing CV units. A CV segment is analyzed frame by frame. Each frame is represented by 12 MFCC coefficients, energy, and their first order and second order derivatives, thus by a 39-dimensional feature vector.

5 Experiments and Results

5.1 HMM Based System Adaptation

In this work, the above mentioned cross-language adaptation methods have been implemented. For each language, the HMM based speech recognition system was built and then adapted to other languages. For building a system for the source language, a 5-state, left-to-right, continuous density HMM using multiple mixtures with a diagonal covariance matrix is trained for each CV unit having at least 50 examples in the data. The performances of the source language systems are, for Tamil 50.5 %, for Telugu 46.5 % and for Hindi 40.1 %.

The speech recognition system for the source language is adapted to the target language by using the limited amount (2 bulletins) of adaptation data. For target language, we built acoustic models for those CV units that have at least 5 examples in the adaptation data. Table 2 shows the performance of different adaptation methods. The results given are the CV unit recognition accuracies and are obtained without using any language models. It is seen that the combined method of adaptation, i.e., MLLR+MAP, performs better than the individual adaptation methods.

Table 3 compares the performance of the MLLR+MAP with the native speech training (NST). Speech recognizers for Tamil and Telugu languages got a small performance improvement when adapted from Telugu and Tamil respectively. When adapted from Hindi, both the systems have shown a reduction in the accuracy. Performance for Hindi language system was reduced when adapted from Tamil or Telugu. The reason is that both Tamil and Telugu languages are Dravidian languages, and Hindi is an Aryan language. It is seen from Table 3

Table 2. Comparison of CV unit recognition accuracies for different adaptation methods

Source language	Target language	Adaptation method			
		Bootstrap	MLLR	MAP	MLLR+MAP
Tamil	Telugu	23.38	23.50	25.50	25.55
Telugu	Tamil	27.69	25.60	29.04	29.20

Table 3. Comparison of CV unit recognition accuracies of native speech training (NST), bootstrap and MLLR+MAP adaptation methods

Target language (No. of CV units)	NST	Adaptation Method	Source language		
			Tamil	Telugu	Hindi
Tamil (110)	25.57	Bootstrap	-	27.69	27.73
		MLLR+MAP	-	29.20	25.86
Telugu (134)	23.64	Bootstrap	23.38	-	23.12
		MLLR+MAP	25.55	-	22.05
Hindi (98)	35.54	Bootstrap	33.41	34.74	-
		MLLR+MAP	29.41	30.64	-

that when a language from one family is adapted from a language in the other family, the bootstrap method performs better than the MLLR+MAP method.

5.2 SVM Based System Adaptation

We built the SVM based recognizers using the native speech training and pooled adaptation. The SVM classifier requires a fixed length pattern representing a CV utterance. For this purpose, the point where the consonant ends and vowel begins in the CV utterance, called vowel onset point (VOP) is detected. Then the five overlapping frames to the left of VOP and five frames to the right of VOP are considered to represent a CV utterance. As before, each frame is represented by a 39-dimensional feature vector. Thus, each CV utterance is now represented by a 390-dimensional feature vector [9]. The performances of the source language systems are, for Tamil 50.1 %, for Telugu 50.6 % and for Hindi 41 %. The results for the SVM based adaptation methods are given in Table 4. The performance of the SVM based system is less in comparison with the HMM based system. The reason for this reduction is that by simply pooling the data of different languages, the confusability among classes will be increased and the SVMs are more effected as they are discriminative in nature. The proposed method that pools support vectors of models in source language system with adaptation data has given approximately the same performance as pooling, with a reduction in the training time by a factor of 4.

We also compared 1-best, 2-best and 5-best results for the native speech training, HMM based MLLR+MAP adaptation and the proposed SVM based

Table 4. Performance of SVM based adaptation methods for 2 bulletin adaptation data

Target language (No. of CV units)	NST	Adaptation Method	Source language		
			Tamil	Telugu	Hindi
Tamil (110)	22.96	Pooling	-	18.71	18.41
		Using SVs	-	22.76	16.86
Telugu (134)	19.29	Pooling	18.54	-	17.58
		Using SVs	18.33	-	16.66
Hindi (98)	30.75	Pooling	25.12	25.39	-
		Using SVs	25.76	24.48	-

Table 5. Comparison of n-best performance for the HMM based systems using the MLLR+MAP adaptation method and the SVM based systems using the proposed adaptation method, for n=1, 2, 5 : B is the number of bulletins used for adaptation

Target language (#Classes)	B	System	Native Speech Training			Source language system								
						Tamil			Telugu			Hindi		
			1 best	2 best	5 best	1 best	2 best	5 best	1 best	2 best	5 best	1 best	2 best	5 best
Tamil (110)	2	SVM	23.0	33.4	49.2	-	-	-	22.8	33.1	48.1	16.9	27.3	44.5
		HMM	25.6	29.7	34.3	-	-	-	29.2	33.6	39.5	25.9	29.9	34.9
	4	SVM	33.1	45.0	61.2	-	-	-	26.5	38.7	55.9	26.7	37.9	54.3
		HMM	39.2	45.6	51.7	-	-	-	32.0	38.0	45.1	33.4	39.1	45.7
Telugu (134)	2	SVM	19.3	27.6	39.5	18.3	27.8	40.9	-	-	-	16.7	24.7	37.6
		HMM	23.6	27.8	32.8	25.5	28.9	35.2	-	-	-	22.0	26.5	32.1
	4	SVM	33.0	43.7	57.8	27.9	39.4	54.8	-	-	-	29.6	40.4	54.9
		HMM	40.0	45.8	52.2	32.8	39.1	46.5	-	-	-	32.0	37.8	45.3
Hindi (98)	2	SVM	30.7	41.6	55.0	25.8	35.5	50.1	24.5	35.0	49.2	-	-	-
		HMM	35.5	39.1	43.7	29.4	33.7	39.5	30.6	34.5	39.6	-	-	-
	4	SVM	28.1	37.9	53.2	26.2	36.5	50.5	25.7	35.1	49.0	-	-	-
		HMM	34.1	37.7	41.9	26.1	30.1	35.5	26.9	31.3	36.2	-	-	-

adaptation methods. These results are given in Table 5. The 2-best and 5-best results of SVM based methods are better than that of HMM based methods.

6 Conclusions

We compared the performance of various cross-language adaptation methods for building speech recognition systems using small amount of data for Indian languages. We proposed a variation of pooling method using support vectors of models in the source language system for SVM based system adaptation and compared it with native speech training and pooling methods. Though it did not improve the performance, it reduced the training time significantly. The SVM based methods performed better than the HMM based methods when the

2-best and 5-best results are considered. The performance of adapted systems with small amount of data is far less than that of corresponding systems built from large amounts of native speech data. The usability of incremental SVM learning for cross-language adaptation will be studied in the future work.

References

1. Waibel, A., Geutner, P., Mayfield Tomokiyo, L., Schultz, T., Woszczyna, M.: Multilinguality in speech and spoken language systems. Proceedings of the IEEE 88(8), 1297–1313 (2000)
2. Uebler, U.: Multilingual speech recognition in seven languages. Speech Communication 35, 53–69 (2001)
3. Leggetter, C.J., Woodland, P.C.: Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. Computer Speech and Language 9(2), 171–185 (1995)
4. Wang, Z., Schultz, T., Waibel, A.: Comparison of acoustic model adaptation techniques on non-native speech. In: Proc. of ICASSP 2003, vol. 1, pp. 540–543 (2003)
5. Schultz, T., Waibel, A.: Language-independent and language-adaptive acoustic modeling for speech recognition. Speech Communication 35(1-2), 31–51 (2001)
6. Zhao, X., O'Shaughnessy, D.: An evaluation of cross-language adaptation and native speech training for rapid HMM construction based on very limited training data. In: Proc. of INTERSPEECH, pp. 1433–1436 (August 2007)
7. Wheatley, B., Kondo, K., Anderson, W., Muthusamy, Y.: An evaluation of cross-language adaptation for rapid HMM development in a new language. In: Proc. of ICASSP 1994, vol. 1, pp. 237–240 (April 1994)
8. Nieuwoudt, C., Botha, E.C.: Cross-language use of acoustic information for automatic speech recognition. Speech Communication 38(8), 101–113 (2002)
9. Gangashetty, S.V., Sekhar, C.C., Yegnanarayana, B.: Extraction of fixed dimension patterns from varying duration segments of consonant-vowel utterances. In: Proc. of ICISIP 2004, January 2004, pp. 159–164 (2004)

Automatic Classification System for the Diagnosis of Alzheimer Disease Using Component-Based SVM Aggregations

I. Álvarez[*], M. López, J.M. Górriz, J. Ramírez, D. Salas-Gonzalez,
C.G. Puntonet, and F. Segovia

Dept. of Signal Theory, Networking and Communications
University of Granada, Spain
{illan,miriamlp,gorriz,javierrp,dsalas,carlos,fsegovia}@ugr.es

Abstract. The early detection of subjects with probable Alzheimer Type Dementia (ATD) is crucial for effective appliance of treatment strategies. Functional brain imaging including SPECT (Single-Photon Emission Computed Tomography) and PET (Positron Emission Tomography) are commonly used to guide the clinician's diagnosis. Nowadays, no automatic tool has been developed to aid the experts to diagnose the ATD. Instead, conventional evaluation of these scans often relies on subjective, time consuming and prone to error steps. This paper shows a fully automatic computer-aided diagnosis (CAD) system for improving the accuracy in the early diagnosis of the ATD. The proposed approach is based on the majority voting cast by an ensemble of Support Vector Machine (SVM) classifiers, trained on a component-based feature extraction technique which searches the most discriminant regions over the brain volume.

1 Introduction

In the interpretation of cerebral functional images such as PET (Positron Emission Tomography) and SPECT (Single Photon Emission Computed Tomography), well-trained classifiers provide effectively diagnostic information. Both SPECT and PET are non-invasive, three-dimensional functional imaging modalities that provide clinical information regarding biochemical and physiologic processes in patients, i.e. the regional blood flow (rCBF). Many studies have examined the predictive abilities of nuclear imaging with respect to ATD and other dementia illnesses. The evaluation of these images is usually done through visual ratings performed by experts. However, maybe due to the large amounts of data represented in comparison with the number of available imaged subjects

[*] This work was partly supported by the Spanish Government under the PETRI DENCLASES (PET2006-0253), TEC2008-02113, NAPOLEON (TEC2007-68030-C02-01) projects and the Consejería de Innovación, Ciencia y Empresa (Junta de Andalucía, Spain) under the Excellence Project (TIC-02566).

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 402–409, 2009.

(typically <100), statistical classification methods have not been widely used in this area.

Support Vector Machines (SVM) are a core machine learning technology [1]. They have strong theoretical foundations and excellent empirical successes. This technique have been successfully applied to many fields including voice activity detection (VAD) [2], content-based image retrieval [3], and medical imaging diagnosis [4]. Combining SVM with other techniques regarding more effective training and decision making approaches [8], some usual problems related to high data dimension and small sample size can be substantially improved.

This paper shows a computer aided diagnosis (CAD) system for the early detection of ATD using SVM classifiers trained on several regions of the brain image, obtained from a previous feature extraction phase. The resulting SVM ensemble is combined under a pasting-votes strategy to give a final decision output on the patient. The proposed method, tested over SPECT and PET images, is developed with the aim of reducing the subjectivity in visual interpretation of these scans by clinicians, thus improving the accuracy of diagnosing Alzheimer disease in its early stage.

2 Background on SVMs

SVMs separate a given set of binary labeled training data with a hyperplane that is maximally distant from the two classes (known as the maximal margin hyperplane). The objective is to build a function $f : R^N \longrightarrow \{\pm 1\}$ using training data, consisting of N-dimensional patterns $\mathbf{x}_i$ and class labels y_i:

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_l, y_l) \in \left(R^N \times \{\pm 1\}\right), \tag{1}$$

so that f will correctly classify new examples $(\mathbf{x}, y)$. When no linear separation of the training data is possible, SVM can work effectively in combination with kernel techniques using the *kernel trick*, so that the hyperplane defining the SVM corresponds to a non-linear decision boundary in the input space. In this way the decision function f can be expressed only in terms of the *support vectors*:

$$f(\mathbf{x}) = \sum_{i=1}^{N_S} \alpha_i y_i K(\mathbf{s}_i, \mathbf{x}) + w_0, \tag{2}$$

where $K(.,.)$ is the kernel function, α_i is a weight constant derived from the SVM process and $\mathbf{s}_i$ are the support vectors [1].

2.1 SVM Ensemble

In SVM ensemble, individual SVMs are aggregated to make a collective decision in several ways such as the majority voting, least-squares estimation-based weighting, and the double layer hierarchical combing [5]. The training SVM ensemble can be conducted in the way of bagging or boosting. In bagging, each

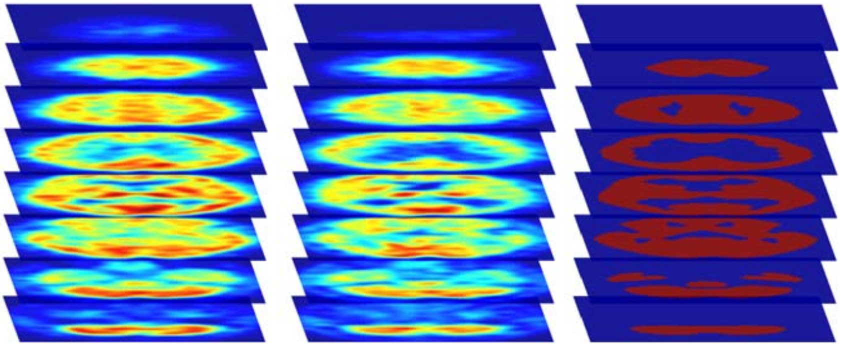


Fig. 1. Cross sections of: *Left column*: A normal patient. *Central column*: A DTA patient. *Right column*: Averaged mask.

individual SVM is trained independently using the randomly chosen training samples via a bootstrap technique. In boosting, each individual SVM is trained using the training samples chosen according to the samples probability distribution that is updated in proportion to the error in the sample. When memory limitations exist, voting many classifiers built on small subsets of data ("pasting small votes") is an approach for learning from massive datasets [8]. SVM ensemble is essentially a type of cross-validation optimization of single SVM, having a more stable classification performance than other models.

In this paper the brain image of each patient is divided into components with several shapes and sizes. These components are obtained after a feature extraction process and then, an individual SVM is applied on each component. This yields to a SVM ensemble for each brain image that can be used to grow a global diagnostic of the patient.

3 Image Preprocessing

As in previous approaches [4] the classification task is based on the assumption that the same position in the volume coordinate system within different volumes corresponds to the same anatomical position. This makes it possible to do meaningful voxel wise comparisons between images. However this assumption is not met by the images without pre-processing: The subject who is being imaged is not always positioned at the same position in the reference frame of the imaging device and the anatomy does not always have the same shape and size between different subjects. This means that registering the volumes spatially is needed. This is done by an implementation of the algorithms proposed in [6]. On the other hand, direct comparison of the voxel intensities between SPECT or PET images, even between different acquisitions of the same subject, is not possible without normalization of the intensities. For all the experiments, we normalize the intensities by applying an affine transformation to the intensities as suggested also in [6]. All the images of the database are transformed using affine and non-linear spatial normalization, thus the basic assumptions are met.

4 Feature Selection

A major problem associated with pattern recognition systems is the so-called *curse of dimensionality*, that is, the number of available features for designing the classifier can be very large compared with the number of available training examples. There are clear motivations for reducing the dimensionality of the feature space to a reasonable minimum: *i*) reduction of the computational cost of the training and testing algorithms, *ii*) elimination of correlation between features, and *iii*) selection of the most discriminant set of features.

Firstly, we construct a binary mask which selects the voxels of interest and discards the rest. This is done by taking the voxels whose mean intensity value averaged over all images exceeds the half of the maximum mean intensity value, and this mask is applied to the original images. In the resulting averaged images the irrelevant information has been removed or reduced. Fig. 1 shows an example of a partial view of different cross sections of a norm subject, a patient with a typical perfusion pattern of ATD and a mask obtained as an average image. Secondly, the brain image obtained in the previous process is divided into several parts or components (each one denoted by c). Such components, which will be used as feature vectors for the SVM classification task, entail a dimensional reduction in the feature space with respect to the Voxels-As-Features (VAF) approximation [4], which we will use as a reference. Note that clinicians only use some image components in the evaluation process of the subject, and this approach aims at finding these parts automatically. These regions are called regions of interest or ROIs.

Two different approaches to the brain division are carried out, depending on the shape of the regions or components that the brain is divided into. The first one divides the brain into components as chains of consecutive voxels. The voxels can be consecutive in three orthogonal directions: coronal, axial or sagittal, so three different implementations will be taken. The number of divisions of the brain will vary from 15 to 25 components, with a consistent variation of the chain length which ranges from 107 to 180 voxels approximately. Note that the dimension of the feature vector has been drastically reduced, so the small size sample problem is lightened. This *elongated* shaped division will be shifted in order to adjust the artificial division to the natural shape of the image. The second way of dividing the brain is thought to improve this adjustment, adapting the ROIs form to the artificial segmentation by giving a *cubic* shape to the components. Also the cubic components will be shifted and their size will be changed. Finally, the component ensemble will serve us to construct a ROIs map.

5 Formation of the Component-Based SVM Ensemble by ROIs

Component-based feature extraction process has actually been applied to the face detection problem [7], centering the components on the eyes, nose and mouth. Our approach does not search for any particular zone, but computes

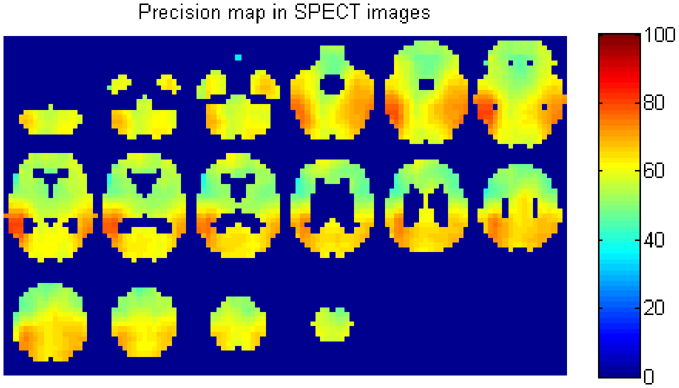


Fig. 2. Map of the Regions of Interest in SPECT images obtained by the values of A_i

a systematic scan of the whole brain image. For the analysis, let M be the number of patients, N the number of components the brain is divided into, and c_{ij}, i=1, 2, ..., N j=1,2, ..., M the component i of the patient j. Therefore, the whole brain image I_j of the patient j is:

$$I_j = c_{1j} \cup c_{2j} \cup c_{3j} \cup ... \cup c_{Mj} \qquad (3)$$

The image[1] I_j will have an associated label y_j, which will be +1 in case the patient is ATD, and −1 in case the patient is NORMAL. Thus, this label y_j is shared with all the components of the image I_j $(c_{1j}, c_{2j}, ..., c_{Mj})$. Each component c_{ij} is used as the feature vector input to train and test a single SVM classifier by means of a Leave-One-Out cross-validation strategy, that is, the classifier is trained on all but one component $\{c_{i1}, c_{i2}, ..., c_{iM-1}\}$, categorizing the remaining component c_{iM} by a label $\ell_{iM} \in \{\pm 1\}$. This process will provide as many SVM classifiers as components the brain is broken down into, being each classifier trained only on its associated region of the brain volume. If the information of a particular region is important with regard to the Alzheimer disease diagnosis, the associated SVM classifier will have a good performance in the classification task. In other words, after the training and testing stages, it is possible to assign an accuracy rate A_i, $i = 1, 2, ..., N$, to each region according to the number of correctly classified patients it provided.

The accuracy values A_i for SPECT images are represented in Fig. 2. The ROIs correspond to the most discriminant components, near red colors, meaning dark red a 100% accuracy in classification. In this figure the value of the accuracy on each voxel is the mean of the different A_i values of every component that contained that voxel during the scan. Fig. 3 shows a diagram of the process steps given in this approach.

[1] Note that $c_{nj} \cap c_{(n+1)j} \neq \emptyset$.

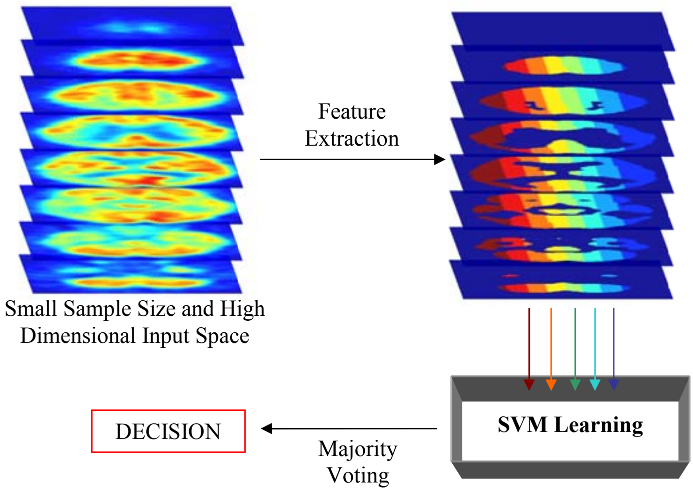


Fig. 3. Steps of the classification process. In the feature extraction stage, both the mask application and the brain division into smaller components face up to the small size sample and high dimensional problems. Each component is used to train an individual classifier which will cast a weighted vote on the patient state. The final decision is taken by the most relevant components majority voting.

5.1 Decision

The SVM ensemble will serve us to define a new decision function based on the pasting-votes technique [8]. The function defined as

$$\mathcal{F}(I_j) = \sum_{i=1}^{N} \ell_{ij} \tag{4}$$

is an non-weighted sum of votes that each component casts and will classify the patient j as NORMAL if $\mathcal{F}(I_j) < 0$, and as ATD if $\mathcal{F}(I_j) > 0$. An improvement is easily introduced by assigning a weight to each vote, which defines a new function:

$$\mathcal{G}(I_j) = \frac{\sum_{i=1}^{N} \ell_{ij} A_i}{\sum_{i=1}^{N} A_i} \tag{5}$$

The definition of a decision function $\mathcal{T}$ based on the non-weighted sum of a limited number S of votes will be determinant. S is chosen by setting a threshold T in the accuracy values: only those components $S \subset I$ whose values A_i are higher than T are allowed to vote:

$$\mathcal{T}(I_j) = \sum_{i \subset S} \ell_{ij}, \quad S = \{i \setminus A_i > T\} \tag{6}$$

As an example, taking T to be 80% will correspond to take S to be the red colored regions in Fig. 2.

Table 1. Accuracy results obtained from SPECT and PET images, considering different shapes and functions

SPECT	**15 Elongated**	**20 Elongated**	**25 Elongated**	**Cubic**
$\mathcal{F}$ **function**	83.5%	86.1%	87.3%	84.8%
$\mathcal{G}$ **function**	87.3%	87.3%	88.6%	86.1%
$\mathcal{T}$ **function** ($T = 87$)	94.9%	93.7%	94.9%	**97.5%**
PET	**15 Elongated**	**20 Elongated**	**25 Elongated**	**Cubic**
$\mathcal{F}$ **function**	96.6%	98.3%	98.3%	93.3%
$\mathcal{G}$ **function**	96.6%	98.3%	98.3%	95%
$\mathcal{T}$ **function** ($T = 97$)	98.3%	98.3%	98.3%	**100%**

6 Evaluation Results

SPECT and PET images used in this work were taken with a PRISM 3000 machine and a SIEMENS ECAT 47 respectively. Initially they were labeled by experienced clinicians of the "Virgen de las Nieves" Hospital (Granada, Spain) and "Clinica PET Cartuja" (Seville, Spain) respectively. The database consists of 79 SPECT patients (41 labeled as NORMAL and 38 labeled as ATD) and 60 PET patients (18 NORMAL and 42 ATD). The dimensionality reduction of the original brain image $79 \times 95 \times 69$ voxel sized was performed by averaging over different sizes of voxels, ranging from $4 \times 4 \times 4$ to $9 \times 9 \times 9$. Different numbers of elongated and cubic components were tested as well, in order to find the optimal values.

Best accuracy values for SPECT images was found when a $7 \times 7 \times 7$ initial averaging was performed for 25 elongated components and a $4 \times 4 \times 4$ voxel sized cubic ones when the $\mathcal{T}$ function is used. For PET images, best results were achieved when a $9 \times 9 \times 9$ initial dimension reduction is applied for 25 elongated components and $4 \times 4 \times 4$ again for cubic components. Making use of the $\mathcal{F}$ function, only one patient, which was initially labeled as NORMAL, is misclassified by the classifier. This misclassification is corrected by the use of a $\mathcal{T}$ function. Clinicians have detected that, although it is a normal patient in the sense that he does not present any sign of ATD, there is a peculiarity in his thalamus metabolism. Different values for T ranging from 50 to 100 were evaluated and finally 97.5% and 100% accuracy values ware attained for SPECT and PET images respectively. Best accuracy results achieved with the corresponding T values are summarized in Table 1, computed for each of the decision functions defined in section 5.1, $\mathcal{F}, \mathcal{G}, \mathcal{T}$. These results outperform the VAF approach, which reaches 78.5% and 96.6% accuracy values for SPECT and PET images respectively.

7 Conclusions

A computer aided diagnosis system for the early detection of the Alzheimer disease was shown in this paper. The system was developed by exploring the masked brain volume in search of discriminant ROIs using different shaped subsets of voxels or components. A single SVM classifier was trained and tested on each component yielding to an ensemble of classification data. These data were aggregated according to a pasting-votes technique by means of three different sum functions. High dependence on the component size was found, contrary to component shape, which had less influence. The best accuracy was obtained for a pasting-vote function that combined the non-weighted sum of votes with the relevant information contained in the ROIs. With this approach, the proposed method provided 97.5% accuracy for SPECT images and a 100% accuracy for PET images, outperforming the reference work results.

References

1. Vapnik, V.N.: Statistical Learning Theory. John Wiley and Sons, Inc., New York (1998)
2. Ramírez, J., Yélamos, P., Górriz, J.M.: SVM-based speech endpoint detection using contextual speech features. Electronics Letters 7(42), 877–879 (2006)
3. Tao, D., Tang, X., Li, X., Wu, X.: Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence 7(28), 1088–1099 (2006)
4. Stoeckel, J., Malandain, G., Migneco, O., Koulibaly, P.M., Robert, P., Ayache, N., Darcourt, J.: Classification of SPECT images of normal subjects versus images of Alzheimer's Disease patients. In: Niessen, W.J., Viergever, M.A. (eds.) MICCAI 2001. LNCS, vol. 2208, pp. 666–674. Springer, Heidelberg (2001)
5. Hyun-Chul, K., Shaoning, P., Hong-Mo, J., Kim, D., Bang, S.Y.: Constructing Support Vector Machine Ensemble. Pattern Recognition 12(36), 2757–2767 (2003)
6. Statistical Parametric Mapping: The Analysis of Functional Brain Images. Academic Press (2007)
7. Huang, J., Blanz, V., Heisele, B.: Face Recognition Using Component-Based SVM Classification and Morphable Models. In: Lee, S.-W., Verri, A. (eds.) SVM 2002. LNCS, vol. 2388, pp. 334–341. Springer, Heidelberg (2002)
8. Breiman, L.: Pasting small votes for classification in large database and on-line. Matching Learning 36, 85–103 (1999)

Early Detection of the Alzheimer Disease Combining Feature Selection and Kernel Machines

J. Ramírez[1], J.M. Górriz[1], M. López[1], D. Salas-Gonzalez[1], I. Álvarez[1],
F. Segovia[1], and C.G. Puntonet[2]

[1] Dept. of Signal Theory, Networking and Communications
University of Granada, Spain
[2] Dept. of Architecture and Computer Technology,
University of Granada, Spain

Abstract. Alzheimer disease (AD) is a progressive neurodegenerative disorder first affecting memory functions and then gradually affecting all cognitive functions with behavioral impairments. As the number of patients with AD has increased, early diagnosis has received more attention for both social and medical reasons. However, currently, accuracy in the early diagnosis of certain neurodegenerative diseases such as the Alzheimer type dementia is below 70% and, frequently, these do not receive the suitable treatment. Functional brain imaging including single-photon emission computed tomography (SPECT) is commonly used to guide the clinician's diagnosis. However, conventional evaluation of SPECT scans often relies on manual reorientation, visual reading and semiquantitative analysis of certain regions of the brain. These steps are time consuming, subjective and prone to error. This paper shows a fully automatic computer-aided diagnosis (CAD) system for improving the accuracy in the early diagnosis of the AD. The proposed approach is based on feature selection and support vector machine (SVM) classification. The proposed system yields clear improvements over existing techniques such as the voxel as features (VAF) approach attaining a 90% AD diagnosis accuracy.

1 Introduction

Alzheimer disease (AD) is the most common cause of dementia in the elderly and affects approximately 30 million individuals worldwide. AD is a progressive neurodegenerative disorder associated with disruption of neuronal function and gradual deterioration in cognition, function, and behavior. With the growth of the older population in developed nations, the prevalence of AD is expected to triple over the next 50 years. The major goals in treating AD currently are to recognize the disease early in order to initiate appropriate therapy and delay functional and cognitive losses. In addition, as powerful antiamyloid therapies are developed, there will be a need to monitor brain changes and treatment efficacy at the earliest stages of the disease, perhaps even in prodromal patients.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 410–417, 2009.

It is hoped that the widespread availability of newer markers will supplement the strengths of the currently available structural and functional imaging techniques in helping to achieve these goals.

During the last years, research in the field of nuclear medical diagnosis by means of brain image tomography has been focused on new modalities or representations which could ease an effective diagnosis and treatment of neurodegenerative diseases such as Alzheimer's disease. Single Photon Emission Computed Tomography (SPECT) is an emission-computed tomography imaging technique that was initially developed in the 1960s, but was not widely used in clinical practice until the 1980s. It is mainly used when structural information is not enough to detect or monitor a functional disorder. Thus, SPECT is a noninvasive, three-dimensional functional imaging modality that provides clinical information regarding biochemical and physiologic processes in patients. A study of the regional cerebral blood flow (rCBF) of the brain is frequently used as a diagnostic tool in addition to the clinical findings. Many studies have examined the predictive abilities of nuclear imaging with respect to AD and other dementing illnesses. The evaluation of these images is usually done through visual ratings performed by experts. However, statistical classification methods have not been widely used in this area, quite possibly due to the fact that images represent large amounts of data and most imaging studies have relatively few subjects (generally <100) [1].

Since their introduction in the late seventies, Support Vector Machines (SVMs) marked the beginning of a new era in the learning from examples paradigm [2]. SVMs have attracted recent attention from the pattern recognition community due to a number of theoretical and computational merits derived from the Statistical Learning Theory [3] developed by Vladimir Vapnik at AT&T. These techniques have been successfully used in a number of applications including voice activity detection (VAD) [4,5], content-based image retrieval [6], texture classification [7] and medical imaging diagnosis [8].

This paper shows a computer aided diagnosis (CAD) system for the early detection of Alzheimer type dementia (ATD) using SPECT images. The proposed method combining SVM concepts and advanced feature extraction schemes is developed with the aim of reducing the subjectivity in visual interpretation of SPECT scans by clinicians, thus improving the accuracy of diagnosing Alzheimer disease in its early stage.

2 Background on Support Vector Machines

SVM separate a given set of binary labeled training data with a hyperplane that is maximally distant from the two classes (known as the maximal margin hyperplane). The objective is to build a function $f : R^N \longrightarrow \{\pm 1\}$ using training data that is, N-dimensional patterns $\mathbf{x}_i$ and class labels y_i:

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_l, y_l) \in R^N \times \{\pm 1\}, \tag{1}$$

so that f will correctly classify new examples $(\mathbf{x}, y)$.

Linear discriminant functions define decision hypersurfaces or hyperplanes in a multidimensional feature space, that is:

$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = 0, \tag{2}$$

where $\boldsymbol{w}$ is known as the weight vector and w_0 as the threshold. The weight vector $\mathbf{w}$ is orthogonal to the decision hyperplane and the optimization task consists of finding the unknown parameters w_i, i= 1, ..., N, defining the decision hyperplane.

Let $\mathbf{x}_i$, i=1, 2, ..., l, be the feature vectors of the training set, X. These belong to either of the two classes, ω_1 or ω_2. When no linear separation of the training data is possible, SVM can work effectively in combination with kernel techniques so that the hyperplane defining the SVM corresponds to a non-linear decision boundary in the input space. If the data is mapped to some other (possibly infinite dimensional) Euclidean space using a mapping $\Phi(\mathbf{x})$, the training algorithm only depends on the data through dot products in such an Euclidean space, i.e. on functions of the form $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. If a "kernel function" K is defined such that $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, it is not necessary to know the Φ function during the training process. In the test phase, an SVM is used by computing dot products of a given test point $\mathbf{x}$ with $\mathbf{w}$, or more specifically by computing the sign of

$$f(\mathbf{x}) = \sum_{i=1}^{N_S} \alpha_i y_i \Phi(\mathbf{s}_i) \cdot \Phi(\mathbf{x}) + w_0 = \sum_{i=1}^{N_S} \alpha_i y_i K(\mathbf{s}_i, \mathbf{x}) + w_0, \tag{3}$$

where $\mathbf{s}_i$ are the support vectors.

3 Image Acquisition and Preprocessing

Patients are comfortably positioned on the imaging couch with the head "immobilized" in a radiolucent head holder. A gamma emitting ^{99m}Tc-ECD radiopharmaceutical is injected and the SPECT scan is acquired by a three-head gamma camera Picker Prism 3000. A total of 180 projections are taken for each patient with a 2-degree angular resolution. Finally, images of the brain cross sections are reconstructed from the projection data using the filtered backprojection (FBP) algorithm in combination with a Butterworth noise removal filter. Such images are 3-dimensional intensity distributions discretized into voxels. Each voxel represents a grey level intensity, which is related to the rCBF pattern of a patient. Each voxel represents a volume of $2.18 \times 2.18 \times 3.56$ mm^3.

The complexity of brain structures and the differences between brains of different subjects make necessary the normalization of the images with respect to a common template [9]. This step allows us to compare the voxel intensities of the brain images of different subjects. The SPECT images are first spatially normalized using the SPM software [10] in order to ensure that the voxels in different images refer to the same anatomical positions in the brain. The normalized method assumes a general affine model with 12 parameters and a cost function

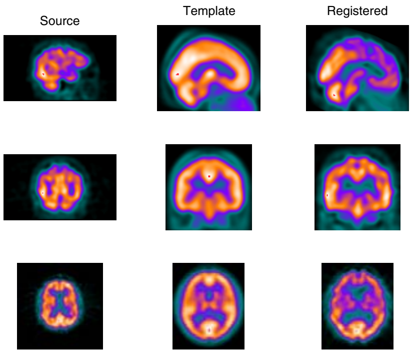


Fig. 1. Three SPECT images. *Left column*: Source image. *Central column*: Template. *Right column*: Transformed image.

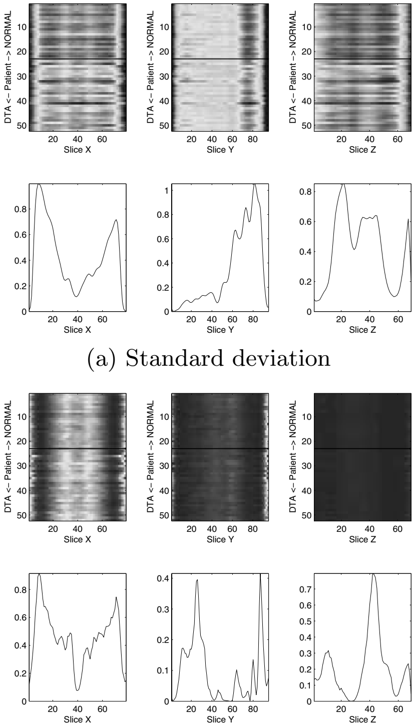
which presents an extreme value when the template and the image are matched together. After the affine normalization, the resulting image is registered using a more complex non-rigid spatial transformation model. The deformations are parameterized by a linear combination of the lowest-frequency components of the three-dimensional cosine transform bases. A small-deformation approach is used and regularization is achieved by the bending energy of the displacement field. Fig. 1 shows an example of the operation of the normalization process on SPECT images. Left column shows arbitrary source images in the dataset, central column shows the template used for image registration, and finally the corresponding normalized images are shown in the right column. It is clearly shown that the transformed image matches the shape of the template.

After the spatial normalization, a $95 \times 69 \times 79$ voxel representation of each subject is obtained. Finally, intensity level of the SPECT images is normalized to the maximum intensity, which is computed for each volume individually by averaging over the 3% of the highest voxel intensities.

4 Feature Selection

A major problem associated with pattern recognition systems is the so-called *curse of dimensionality*, that is, the number of available features for designing the classifier can be very large compared with the number of available training examples. There are clear motivations for reducing the dimensionality of the feature space to a reasonable minimum: *i*) reduction of the computational cost of the training and testing algorithms, *ii*) elimination of correlation between features, and *iii*) selection of the most discriminant set of features.

Features resulting from the first-order statistics provide information related to the intensity level distribution of the image, but they do not give any information about relative positions of the various intensity levels within the image. This information can be extracted from the second-order statistics, where the pixels are considered in pairs. A study was carried out in order to select the best



(a) Standard deviation

(b) Correlation

Fig. 2. Standard deviation and correlation of sagittal, coronal and transversal sections and Fisher discriminant ratio

discriminant set of features of the AD. The analysis considered first- and second-order statistics of sagittal, coronal and transversal slices of the brain. The Fisher linear discriminant ratio defined by:

$$FDR = \frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 + \sigma_2^2} \tag{4}$$

was used as class separability measure where μ_1 and μ_2 denote the with-in class mean value of the feature and σ_1^2 and σ_2^2 their variances.

Among all the features evaluated, the standard deviation and correlation were found to be the most discriminant features of the Alzheimer disease. Fig. 2 shows a plot of the standard deviation and correlation for all the patients in the dataset and all the x, y and z slices corresponding to the sagittal, coronal and transversal views of the brain, respectively. The first row of each figure represents the feature value for each patient and all the slices. Note that, normal and ATD subjects are grouped and separated by an horizontal black line to easily observe the differences among the two classes. It is expected that a careful selection of the region of interest influenced by a discrimination analysis can improve the classifier effectiveness significantly. The value of the Fisher linear discriminant

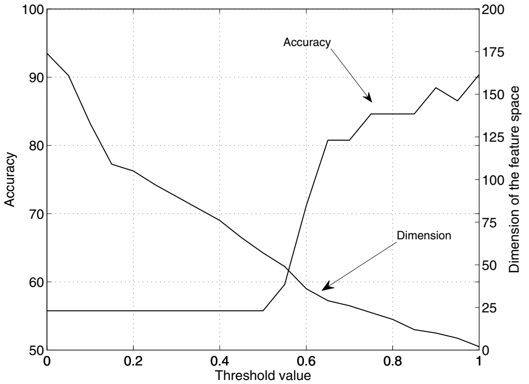


Fig. 3. Accuracy and dimension of the feature space for a RBF SVM system trained using the standard deviation and correlation of the slices with the normalized FDR exceeding a given threshold

ratio is also plotted below each set of features. It can be concluded that not all the slices in the volume element provide the same discriminant value. Moreover, the standard deviation for the y slices and the correlation of the x slices are the most discriminant features.

5 Evaluation Results

SPECT images used in this work were initially labeled by experienced clinicians of the "Virgen de las Nieves" Hospital (Granada, Spain) using four different labels: normal (NOR) for patients without any symptoms of ATD and possible ATD (ATD-1), probable ATD (ATD-2) and certain ATD (ATD-3) to distinguish between different levels of presence of typical symptoms of ATD. The database consists of 52 patients: 23 NOR, 13 ATD-1, 12 ATD-2 and 4 ATD-3. We combine the latter three labels and only use two classes: NOR and ATD.

Aiming at reducing the dimensionality of the feature space and further improving the performance of the CAD system by means of more effective kernels, a SVM-based classifier was developed using the most discriminant set of features: standard deviation of coronal slices and correlation of sagittal slices. Dimensionality of the feature space is reduced by considering only the features of the slices with normalized Fisher linear discrimination ratio exceeding a threshold. Fig. 3 shows the accuracy of the CAD system and the dimension of the feature vector as a function of the threshold value when a RBF kernel is used. Note that the accuracy of the system increases up to 90% as the threshold increases. The best results are obtained for a two-dimensional feature vector consisting of the standard deviation and correlation of the coronal and sagittal slices with the highest value of the Fisher discriminant ratio as shown in figure 2. Note that these results are in agreement with the SVM concept. In high dimensional

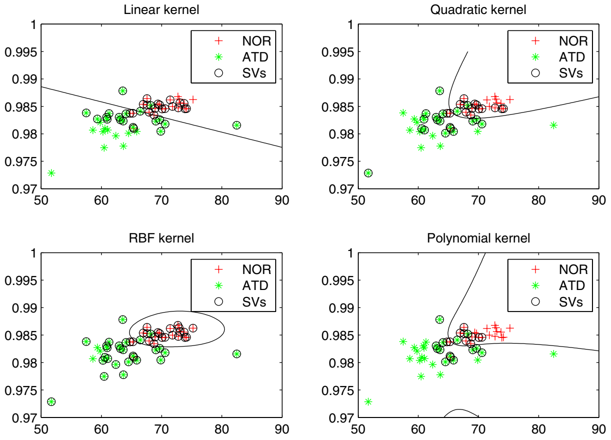


Fig. 4. Decision functions in the input space for a two-dimensional feature vector consisting of the most discriminant coronal standard deviation and sagittal correlation slices

feature spaces, RBF kernels perform poorly. Meanwhile, reducing the dimensionality of the feature space by selecting the most discriminant slices in the volume improves the accuracy of the system. The benefits are obtained as a result of mapping the low-dimensional input space into a high-dimension feature space where the data becomes linearly separable.

Fig. 4 shows the training patterns, their associated class labels as well as the support vectors defining the SVM classification rule when linear, quadratic, RBF and polynomial kernels are used for mapping into the feature space. It is clearly shown that reducing the dimensionality of the input space to a two-coefficient feature space yields high discrimination accuracy. Among all the experiments carried out, RBF kernel functions yielded the best results with a 90.38% classification accuracy. Meanwhile, linear kernels, that achieve the best results in a high dimension input space such as in a voxel-as-feature (VAF) approach [8], yielded just a 84.62% classification accuracy. Thus, the proposed features yielded benefits over the VAF approach where the high dimension of the input space makes unnecessary a non-linear mapping into the feature space as well as linear SVM the most practical [8] over quadratic, RBF and polynomial kernels.

6 Conclusions

A computer aided diagnosis system for the early detection of the Alzheimer disease was shown in this paper. The system was developed by exploring the most discriminant set of features among first- and second-order statistics of the human brain. Moreover, reducing the dimensionality of the input space to a two-coefficient feature vector yielded high discrimination accuracy specially when a RBF kernel is used. The experiments showed that the proposed features yielded significant improvements over recently developed VAF approaches where the high dimension of the input space makes linear SVM the most effective when compared to quadratic, RBF and polynomial kernels. With these and other innovations, the proposed method provided a 90.38% accuracy for the early diagnosis of the ATD.

Acknowledgments. This work was partly supported by the MICINN under the PETRI DENCLASES (PET2006-0253), TEC2008-02113, NAPOLEON (TEC2007-68030-C02-01) and HD2008-0029 projects and the Consejería de Innovación, Ciencia y Empresa (Junta de Andalucía, Spain) under the Excellence Project (TIC-02566).

References

1. Ishii, K., Kono, A.K., Sasaki, H., Miyamoto, N., Fukuda, T., Sakamoto, S., Mori, E.: Fully automatic diagnostic system for early- and late-onset mild Alzheimer's disease using FDG PET and 3D-SSP. European Journal of Nuclear Medicine and Molecular Imaging 33(5), 575–583 (2006)
2. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2(2), 121–167 (1998)
3. Vapnik, V.N.: Statistical Learning Theory. John Wiley and Sons, Inc., New York (1998)
4. Enqing, D., Guizhong, L., Yatong, Z., Xiaodi, Z.: Applying support vector machines to voice activity detection. In: 6th International Conference on Signal Processing, vol. 2, pp. 1124–1127 (2002)
5. Ramírez, J., Yélamos, P., Górriz, J.M., Segura, J.C.: SVM-based speech endpoint detection using contextual speech features. Electronics Letters 42(7), 877–879 (2006)
6. Tao, D., Tang, X., Li, X., Wu, X.: Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(7), 1088–1099 (2006)
7. Kim, K.I., Jung, K., Park, S.H., Kim, H.J.: Support vector machines for texture classification. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(11), 1542–1550 (2002)
8. Fung, G., Stoeckel, J.: SVM feature selection for classification of SPECT images of Alzheimer's disease using spatial information. Knowledge and Information Systems 11(2), 243–258 (2007)
9. Salas-González, D., Górriz, J.M., Ramírez, J., Lassl, A., Puntonet, C.G.: Improved gauss-newton optimization methods in affine registration of SPECT brain images. IET Electronics Letters 44(22), 1291–1292 (2008)
10. Friston, K.J., Ashburner, J., Kiebel, S.J., Nichols, T.E., Penny, W.D.: Statistical Parametric Mapping: The Analysis of Functional Brain Images. Academic Press, London (2007)

Computer Aided Diagnosis of Alzheimer Disease Using Support Vector Machines and Classification Trees

D. Salas-Gonzalez[1], J.M. Górriz[1], J. Ramírez[1],
M. López[1], I. Álvarez[1], F. Segovia[1], and C.G. Puntonet[2]

[1] Dept. of Signal Theory, Networking and Communications, University of Granada,
18071 Granada, Spain
dsalas@ugr.es, gorriz@ugr.es, javierrp@ugr.es

[2] Dept. of Computer Architecture and Computer Technology, University of Granada,
18071, Granada, Spain
carlos@atc.ugr.es

Abstract. This paper presents a computer-aided diagnosis technique for improving the accuracy of the early diagnosis of the Alzheimer type dementia. The proposed methodology is based on the combination of support vector machine learning with linear kernels and classification trees. The classification tree technique allows to choose wisely the most discriminant set of voxels in the images. Thus, the classification tree produces a considerably improvement upon considering the support vector machine classifier only.

1 Introduction

Distinguishing Alzheimer disease remains a diagnostic challenge specially during the early stage of the disease. Furthermore, in this early stage, the disease offers better opportunities to be treated.

Single photon emission computed tomography (SPECT) is a widely used technique to study the functional properties of the brain.

SPECT brain imaging techniques employ radioisotopes which decay emitting predominantly a single gamma photon. When the nucleus of a radioisotope disintegrates, a gamma photon is emitted with a random direction which is uniformly distributed in the sphere surrounding the nucleus. If the photon is unimpeded by a collision with electrons or other particles within the body, its trajectory will be a straight line or ray. In order for a photon detector external to the patient to discriminate the direction that a ray is incident from, a physical collimation is required. Typically, lead collimator plates are placed prior to the the detectors crystal in such a manner that the photons incident from all but a single direction are blocked by the plates. This guarantees that only photons incident from the desired direction will strike the photon detector. SPECT has become an important diagnostic and research tool in nuclear medicine.

In SPECT images, the differences between different brains make necessary the normalization of the images with respect to a reference template. The general

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 418–425, 2009.

affine model, with 12 parameters, is usually chosen as a first step in a normalization algorithm before to proceed with a more complex non-rigid spatial transformation model [1, 2].

This paper shows a computer aided diagnosis (CAD) system for the early detection of Alzheimer Type Dementia (ATD) using Support Vector Machines (SVM) classifiers applied to different components of the brain image. SVM is a powerful tool which has been recently focused the attention for the classification of tomography brain images [3, 4, 5]. The information obtained from the components of the image is analyzed under a classification tree approach. The proposed methodology allows us to choose the most relevant components of the image or regions of interest. Furthermore, the classification tree improves the accuracy of the diagnosis upon the use of support vector machines only.

This work is organised as follows: in section 2 we present an overview of the support vector machines and the classification trees; in Section 3, the material and methods are presented; Section 4 contains the results; and, finally, the conclusions are drawn in Section 5.

2 Background

2.1 Support Vector Machines

Support vector machines is a powerful mathematical tool that is able to separate a set of binary labelled training data with a hyperplane which is maximally distant from the two classes. The objective is to build a function $f : R^N \rightarrow \{\pm 1\}$ using training data consisting of N-dimensional patterns $\mathbf{x_i}$ and class labels y_i:

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_l, y_l) \in (R^N \times \{\pm 1\}), \tag{1}$$

so that f will correctly classify new examples $(\mathbf{x}, \mathbf{y})$.

Linear discriminant functions define decision hypersurfaces or hyperplanes in a multidimensional feature space:

$$g(\mathbf{x}) = \mathbf{w^T x} + \mathbf{w_0} = \mathbf{0} \tag{2}$$

where $\mathbf{w}$ is the weight vector and w_0 is the threshold. $\mathbf{w}$ is orthogonal to the decision hyperplane. The goal is to find the unknown parameters $w_i, i = 1, ..., N$ which define the decision hyperplane.

Let $\mathbf{x_i}$, $i = 1, 2, ..., l$ be the feature vectors of the training set X. These belong to two different classes, ω_1 or ω_2. If the classes are linearly separable, the objective is to design a hyperplane that classifies correctly all the training vectors. This hyperplane is not unique and it can be estimated maximizing the performance of the classifier, that is, the ability of the classifier to operate satisfactorily with new data. The maximal margin of separation between both classes is a useful design criterion. Since the distance from a point $\mathbf{x}$ to the hyperplane is given by $z = |g(\mathbf{x})|/ \parallel \mathbf{w} \parallel$, the optimization problem can be reduced to the maximization of the margin $2/ \parallel \mathbf{w} \parallel$ with constraints by scaling $\mathbf{w}$ and w_0 so that the value of $g(\mathbf{x})$ is $+1$ for the nearest point in w_1 and -1 for the nearest point in w_2. The constraints are the following:

$$\mathbf{w^T x + w_0 \geq 1}, \forall \mathbf{x} \in \mathbf{w_1} \tag{3}$$

$$\mathbf{w^T x + w_0 \leq 1}, \forall \mathbf{x} \in \mathbf{w_2}, \tag{4}$$

or, equivalently, minimizing the cost function $J(\mathbf{w}) = 1/2\|\mathbf{w}\|^2$ subject to:

$$y_i(\mathbf{w}^T\mathbf{x}_i + w_0) \geq 1, i = 1, 2, ..., l. \tag{5}$$

2.2 Classification and Regression Trees

Classification trees is a non-parametric technique that produces classification of categorical dependent variables [6]. Binary tree structured classifiers are constructed by repeated splits of subsets of X into two descendant subsets, beginning with X itself. This process is plotted in Figure 1 for a two classes tree. In the figure, X_2 and X_3 are disjoint with $X = X_2 \cup X_3$. Similarly, X_4 and X_5 are disjoint with $X_4 \cup X_5 = X_2$. Those subsets which are not split, in this case X_4, X_5, X_6 and X_7 are called terminal nodes. Each terminal node is designated by a class label. There may be more than one terminal subset with the same class label.

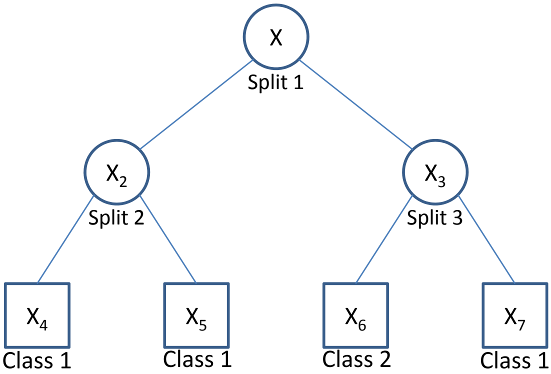


Fig. 1. Hypothetical two-class tree

The first problem in tree construction is how to use the learning sample to determine the binary splits of X into smaller pieces. In order to build a classification tree, three questions need to be solved: how to select the splits, when to declare a node terminal or split and how to assign a class to each terminal node. Once a good split of X is found, then a search is made for good splits of each of the two descendant nodes. More information about how to use data to construct classification (and regression) trees can be found in [6].

3 Material and Methods

3.1 Image Acquisition and Preprocessing

79 Image files were taken from a concurrent study investigating the use of SPECT as a diagnostic tool for the early onset of Alzheimer-type dementia. SPECT data were acquired by a three head gammacamera Picker Prism 3000. The patient is injected with a ^{99m}Tc-ECD radiopharmaceutical which emits gamma rays that are detected by the detector. A total of 180 projections were taken with a 2-degree angular resolution. Images of the cross sections of the brain were reconstructed from the projection data using filtered backprojection (FBP) algorithm in combination with a Butterworth noise removal filter. Finally, a spatial normalization process including affine and non-rigid transformations is performed in order to correct the differences in position and angle of a subject with respect to the SPECT camera in different acquisitions as well as to make anatomies of different subjects correspond, respectively. The result is a 95x69x79 spatially normalized voxel representation of each subject.

After that, the images are resampled to a smaller size (17x23x19). Furthermore, we construct a mask which selects the voxels whose mean intensity value averaged over all images exceeds the half of the maximum mean intensity value. These preprocessing steps reduce the number of voxels of the image.

3.2 SVM Classification

Firstly, we divide the brain image into several parts or components (each one denoted by c).

We divide the brain into components as chains of consecutive voxels. The voxels can be consecutive in three orthogonal directions; coronal, axial or sagittal, so three different implementations will be taken. The number of divisions of the brain will vary from 15 to 25 components, with a consistent variation of the chain length. The elongated shaped division will be shifted in order to adjust the artificial division to the natural shape of the image.

These components will be used as feature vectors for the SVM classification task. Figure 2 shows different cross sections of a normal subject, an image of the brain of a patient with a typical perfusion pattern of disease type Alzheimer and the mask obtained as an average image.

The proposed methodology computes a systematic scan of the whole brain image. For the analysis, let M be the number of patients, N the number of components the brain is divided into, and c_{ij} , $i = 1, 2, ..., N$ $j = 1, 2, ..., M$ the component i of the patient j, the whole brain image I_j of the patient j is:

$$I_j = c_{1j} \cup c_{2j} \cup ... \cup c_{Mj} \tag{6}$$

Each image I_j has an associate label y_j, which is chosen to be 0 for Normal patients and +1 for ATD. These images labels have been labelled by experienced clinicians of the *Virgen de las Nieves* hospital in Granada, Spain. Thus, the label y_j is shared with all the components of the image I_j. Each component is

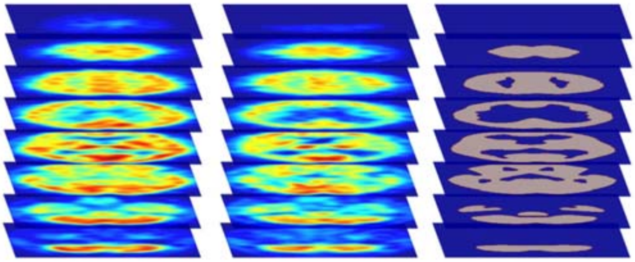


Fig. 2. Left column: A normal patient. Central column: An ATD patient. Right column: Averaged mask.

used as a feature vector input, therefore a SVM classifier with linear kernel is trained and tested on each component c_{ij} using a leave-one-out cross validation strategy. The classifier is trained in all but one component, categorizing the remaining component by a label $\ell_{ij} \in \{$ Normal,ATD$\}$. After processing the whole database, the outcome is an ensemble of individual SVM results which will be used to construct a tree classifier.

3.3 Construction of the Tree Classifier

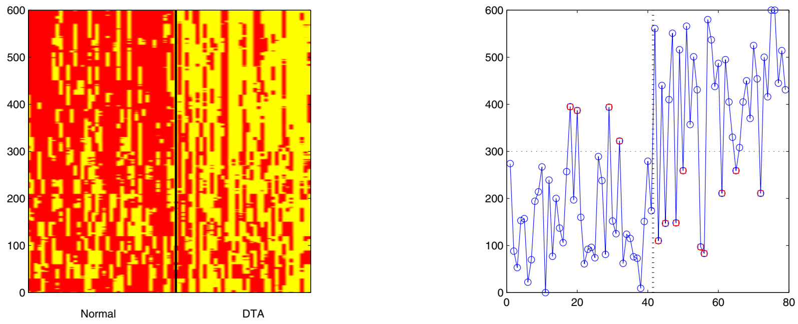
The second step of this work consists in using the learning sample given by the results of the component-based SVM methodology to construct a tree classifier. The training sample consists of data $\{c_{ij}, \ell_{ij}\}$. This classification tree will allow us to choose which components to use in the classification problem.

4 Results

In Figure 3(a), the classification obtained using support vector machines with linear kernel for each of the components of the brain is plotted. 79 SPECT images are placed in the x-axis (41 Normals and 38 ATD). The abscissa presents each of the components which has been classified using SVM and a leave-one-out methodology. Different colours have been used for Normal patients and ATD. Let see that, roughly, a different colour density has been obtained for Normals and ATD patients.

Figure 3(b) depicts the number of components which were found to be ATD by the component-based SVM. A decision whether an image is Normal or ATD can be given using this information. If more than half of the components in an image is classified as Normal (or ATD), this image is automatically labelled as Normal (or ATD) by the SVM classifier. This procedure give us an 84 % accuracy in classification.

A more efficient way to select components for classifying the images can be designed using classification trees [6]. The classification tree is designed to help us to choose which sequence of components is optimal to consider them in the classification task. Therefore, this can be seen as a procedure to select regions of



(a) (b)

Fig. 3. a) Classification results for each of the 600 components using component-based SVM with linear kernel. Dark colour = Normal. Light colour = ATD. b) Number of components which were found to be ATD using the linear SVM classifier for each image. Rectangles: Missclassified images.

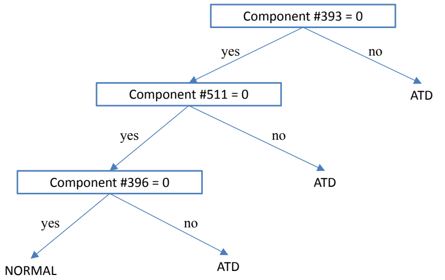
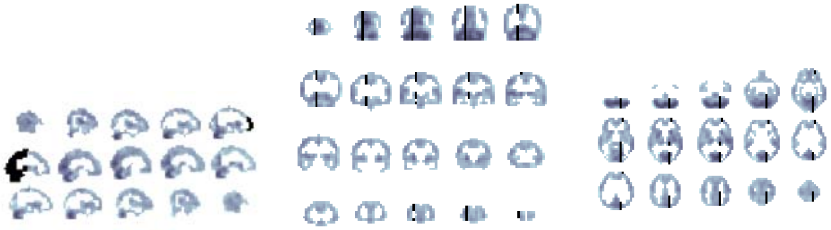


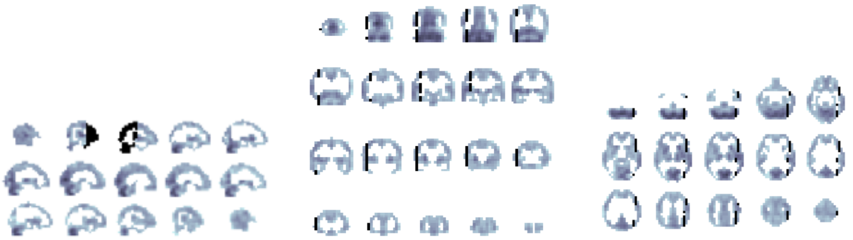
Fig. 4. Decision tree



(a) Sagittal plane. (b) Coronal plane. (c) Axial plane.

Fig. 5. Black: component #393

interest automatically. Figure 4 shows the decision tree which has been obtained for the 79 SPECT images studied. This tree gives us a procedure to increase the accuracy of the SVM classifier up to 90 %. The proposed classifier consists in consider the component-based SVM decision using only the component #393 of the brain. If the classification result is not equal to 0 the image is ATD, otherwise, the component #511 is used in the classification. Again, if the classification result



(a) Sagittal plane. (b) Coronal plane. (c) Axial plane.

Fig. 6. Black: component #511

Table 1. Accuracy rate using only Support Vector Machines with linear kernel (SVM) and combining SVM with the construction of a Classification Tree (SVM-CT)

	SVM	SVM-CT
Accuracy	84%	**90%**

is not equal to 0 the image is ATD, and otherwise, the decision must be taken studying the component #396.

Figures 5 and 6 show the spatial localization in the brain of components selected by the first and second node of the tree structured classifier.

Table 1 gives the accuracy rate for the component-based support vector machine methodology (SVM) and combining the latter with the construction of a classification tree (SVM-CT). It can be seen that the classification tree improves the classification task.

5 Conclusion

In this work, a computer aided diagnosis system for detection of the Alzheimer disease was presented. We put forward a methodology to classify whether a SPECT image of the brain corresponds to a normal or a patient with Alzheimer disease. The proposed methodology is based on the application of support vector machines to different set of voxels of the brain. The regions of interest are computed constructing a classification tree based on the information given by the SVM classification. The proposed methodology has been tested in 79 real single photon emission computer tomography brain images initially labelled by clinicians as 41 normals and 38 Alzheimer type dementia. The classification tree allows to improve the classification accuracy considerably. Improving from 84 % of accuracy using SVM to 90 % combining SVM and classification trees.

Acknowledgment

This work was partly supported by the Spanish MICINN under the PETRI DENCLASES (PET2006-0253), TEC2008-02113, NAPOLEON

(TEC2007-68030-C02-01) projects and the Consejería de Innovación, Ciencia y Empresa (Junta de Andalucía, Spain) under the Excellence Project (TIC-02566).

References

[1] Salas-Gonzalez, D., Górriz, J.M., Ramírez, J., Lassl, A., Puntonet, C.G.: Improved gauss-newton optimization methods in affine registration of spect brain images. IET Electronics Letters 44(22), 1291–1292 (2008)

[2] Friston, K., Ashburner, J., Kiebel, S., Nichols, T., Penny, W. (eds.): Statistical Parametric Mapping: The Analysis of Functional Brain Images. Academic Press, London (2007)

[3] Ramírez, J., Górriz, J.M., Romero, A., Lassl, A., Salas-Gonzalez, D., López, M., Gómez-Río, M., Rodríguez, A.: Computer aided diagnosis of alzheimer type dementia combining support vector machines and discriminant set of features. Accepted in Information Sciences (2008)

[4] Górriz, J.M., Ramírez, J., Lassl, A., Salas-Gonzalez, D., Lang, E.W., Puntonet, C.G., Álvarez, I., López, M., Gómez-Río, M.: Automatic computer aided diagnosis tool using component-based svm. In: Medical Imaging Conference. IEEE, Dresden (2008)

[5] Fung, G., Stoeckel, J.: SVM feature selection for classification of SPECT images of Alzheimer's disease using spatial information. Knowledge and Information Systems 11(2), 243–258 (2007)

[6] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and regression trees. Chapman & Hall, Boca Raton (1993)

Modeling and Prediction of Nonlinear EEG Signal Using Local SVM Method

Lisha Sun, Lanxin Lin, and Chunhao Lin

College of Engineering, Shantou University, Guangdong 515063, China
lssun@stu.edu.cn

Abstract. Electroencephalogram (EEG) is widely regarded as chaotic signal. Modeling and prediction of EEG signals is important for many applications. The method using support vectors machine (SVM) based on the structure risk minimization provides us an effective way of learning machine. The performance of SVM is much better than the traditional learning machine. Now the SVM is used in classification and regression. But solving the quadratic programming problem for training SVM becomes a bottle-neck of using SVM because of the long time of SVM training. In this paper, a local-SVM method is proposed for predicting the signals. The local method is presented for improving the speed of the prediction of EEG signals. The simulation results show that the training of the local-SVM obtains a good behavior. In addition, the local SVM method significantly improves the prediction precision.

1 Introduction

The EEG signals are very complicated pseudo-random signals and serve as windows for us to understand the cerebral activities because these signals are the synthetical reflection of the electricity activities of cerebral tissue and brain function status. EEG plays a more and more important role in the study of brain mechanism and the clinical manifestations of brain diseases with the development of computer and signal processing technology. For example the analysis of cerebral diseases attack can help people to better understand the pathophysiological and pharmacological basis, which is vital in the treatment of cerebral disease and providing useful information for clinic application. The most significant example is epilepsy detection and prediction [1]. It is generally regarded that the Lyapunov exponent of EEG signal varies when the epilepsy comes. At that time, the orderliness of signal is stronger and the chaos is weaker. If we can predict the variation of EEG signals, high-risk operation can be avoided when cerebral diseases happen to the patient [2].

In fact, EEG signal has the characteristics of chaos [1], i.e. EEG signal is a nonlinear spatiotemporal chaotic sequence. According to the Takens's embedded theorem [3], we can reconstruct phase space as long as we choose the appropriate parameters of time delay and embedding dimension. Then many nonlinear prediction techniques based on Volterra system [4], adaptive rational function filter [5], fuzzy logics [6] and learning machine [7] can be used to predict the EEG signal. The learning machine is a common technique for modeling and predicting the chaotic signal. In this paper, we are concerned with prediction performance of the learning machine.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 426–433, 2009.

The traditional learning machines, such as BP neural network and RBF neural network, are based on the empirical risk minimization. In the case of finite training samples, there is a contradiction between the training precision and the generalization. Sometimes reducing the training precision would increase the prediction risk because of the over learning. Recent years, support vector machine (SVM), proposed by Vapnik [8] according to statistical theory, become a hot point and is widely used in classification and regression [9], [10]. SVM applies the structure risk minimization instead of the empirical risk minimization. So it has better characteristic feature of generalization, global optimization and sparse solution. SVM avoids the method selection and over learning problems effectively and solves the problems of non-linear, dimensionality curse and local minimum efficiently.

However in practice, it is of computational complexity to solve a linearly constrained convex quadratic programming problem for training an SVM. Training global SVM will meet great obstacle when the number of the training samples is large. To solve this problem, combining with some other optimization algorithms, we present a new method based on the idea of local method, namely the local SVM. This new method inherits characters of local method which has the advantage of small samples, simplicity and high precision [11]. Combined with these characters, an accurate high-speed prediction method is expected to predict the EEG signals.

This paper is organized as follows. In section 2, the ε-SVM for the prediction of chaotic signal was introduced. In the section 3, a local method was proposed to improve the training speed of SVM in prediction. In section 4, the proposed method was applied to the Logistic chaotic sequence and real EEG signal. Conclusion was given in section 5.

2 ε-SVM for Sequence Prediction

Assuming the finite measured data samples $(\mathbf{x}_1, y_1),\cdots,(\mathbf{x}_l, y_l)\in(\mathbf{X}\times R)$ were obtained from a sample set $P(\mathbf{x}, y)(\mathbf{x}\in R^m, y\in R)$, which follows a certain distribution. The regression of support vector machine is to find a real function $f(\mathbf{x})=\mathbf{w}\bullet\phi(\mathbf{x}_i)+b$ to fit these samples that make the risk function $R[f]=\int c(\mathbf{x}, y, f)dP(\mathbf{x}, y)$ minimum. Where c is the loss function. The error between the observed y and prediction $f(\mathbf{x})$ could be measured by a so called ε insensitive loss function described by equation (1):

$$\left|y_i - f(\mathbf{x}_i,\mathbf{x})\right|_\varepsilon = \max\left\{0,\left|y_i - f(\mathbf{x}_i,\mathbf{x})\right|-\varepsilon\right\} \tag{1}$$

In other words, it may allow some errors. In most cases, the probability density $P(\mathbf{x}, y)$ is not known. It can’t make the risk function minimum directly. Therefore the minimum problem of the following equation (2) is proposed to substitute the risk function.

$$E(\mathbf{w}) = \frac{1}{2}(\mathbf{w}\bullet\mathbf{w}) + C\frac{1}{l}\sum_{i=1}^{l}\left|y_i - f(\mathbf{x}_i,\mathbf{x})\right|_\varepsilon \tag{2}$$

Where $\left|y_i - f(\mathbf{x}_i, \mathbf{x})\right|_\varepsilon = \max\left\{0, \left|y_i - f(\mathbf{x}_i, \mathbf{x})\right| - \varepsilon\right\}$ is the ε insensitive loss function. The first term of right equation (2) represents the complexity of $f(\mathbf{x})$, the second term represents the loss. C represents the compromise relationship between complexity and loss. It equivalents to

$$\begin{cases} \min\limits_{w,\xi_i,\xi_i^*,b} & \dfrac{1}{2}(\mathbf{w}\cdot\mathbf{w}) + C\dfrac{1}{l}\sum\limits_{i=1}^{l}(\xi_i + \xi_i^*) \\ s.t. & \left(\mathbf{w}\cdot\phi(\mathbf{x}_i) + b\right) - y_i \le \varepsilon + \xi_i \\ & y_i - \left(w\cdot\phi(\mathbf{x}_i) + b\right) \le \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \ge 0 \end{cases} \tag{3}$$

It can yield the dual optimization problem:

$$\begin{cases} \max\limits_{\alpha,a^*} & \sum\limits_{i=1}^{l}\left[\alpha_i^*(y_i - \varepsilon) - \alpha_i(y_i + \varepsilon)\right] \\ & -\dfrac{1}{2}\sum\limits_{i=1}^{l}\sum\limits_{j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_i^*)K(\mathbf{x}_i, \mathbf{x}_j) \\ s.t. & \sum\limits_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \quad 0 \le \alpha_i, \alpha_i^* \le C/l, i = 1\cdots l \end{cases} \tag{4}$$

Where $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)\phi(\mathbf{x}_j)$ is kernel. The linear kernels $K(x, y) = xy$, polynomial kernels $K(x, y) = (xy + 1)^d$ and RBF kernels $K(x, y) = \exp(-\|x - y\|_2^2 / \sigma^2)$ are commonly used. From equ.(4), we can get the optimal solutions of α_i and α_i^*, denoted by $\bar{\boldsymbol{\alpha}} = [\bar{\alpha}_1, \bar{\alpha}_1^*, \bar{\alpha}_2, \bar{\alpha}_2^*, \cdots, \bar{\alpha}_l, \bar{\alpha}_l^*]^T$. Then the dynamic system model can be obtained by solving the dual optimization problem. For the input vector $\mathbf{x}$, the prediction can be deduced from:

$$f(\mathbf{x}) = \mathbf{w}\cdot\phi(\mathbf{x}) + \bar{b} = \sum_{i-1}^{l}(\bar{\alpha}_i - \bar{\alpha}_i^*)K(\mathbf{x}_i, \mathbf{x}) + \bar{b} \tag{5}$$

Where $\bar{b}$ can be gotten by equ.(6) or (7). If $\bar{\alpha}_j$ is chosen, then

$$\bar{b} = y_j - \sum_{i=1}^{l}(\bar{\alpha}_i^* - \bar{\alpha}_i)K(\mathbf{x}_i, \mathbf{x}_j) + \varepsilon \quad \bar{\alpha}_j \in (0, C/l) \tag{6}$$

If $\bar{\alpha}_k^*$ is chosen, then

$$\bar{b} = y_k - \sum_{i=1}^{l}(\bar{\alpha}_i^* - \bar{\alpha}_i)K(\mathbf{x}_i, \mathbf{x}_k) - \varepsilon \quad \bar{\alpha}_k^* \in (0, C/l) \tag{7}$$

3 Local ε-SVM

When the samples become large, training the SVM has become the bottle-neck of SVM application. Training algorithms for support vector machines is a research hotspot at present and new algorithms are proposed incessantly. Colin Campbell made an overview of existing training algorithms [12]. In that overview he mainly introduced the SVM light decomposition algorithms [13], sequential minimal optimization (SMO) algorithms proposed by Platt [14], the nearest neighbor algorithms proposed by Kerrthi [15] and the least squares support vector machine algorithms (LS-SVM) proposed by J. A. K. Suykens [16]. These algorithms aimed to improve the SVM training speed in the application of classification or regression.

Sequential minimal optimization is based on the decomposition. It iteratively selects two points and optimizing the target function with respect to them. Then the optimization problem becomes an analytic solution, so the problem of solving the quadratic programming is avoided. Although it needs more times of iteration, but the iteration spends less computing time. Therefore the total time consumed is reduced. The advantage of SMO is that there is no need to store the kernels matrix as well as use quadratic programming package. SMO algorithm will be used in this paper.

Combining with the local method, this paper proposes a local ε-SVM. In the prediction process, the local method selects the vectors which are close to the target vector. Meanwhile, it uses these vectors to train the SVM by SMO algorithm. This increases the training speed because of the little samples. The following part introduces using the local ε-SVM for one-step prediction in chaotic signal.

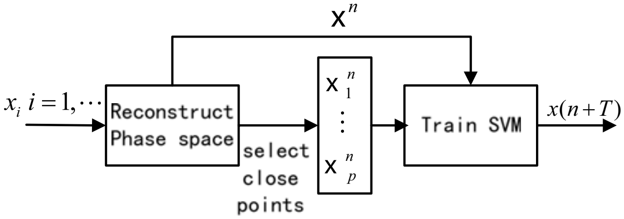


Fig. 1. Local SVM for prediction

1. For a chaotic sequence $x(t)$, select embedding dimension m and delay time τ, reconstruct phase space according to the Takens's embedding theorem. For the final vector $\mathbf{X}(N)$ which is used as the input target vector to predict the next point of the sequence, compute the distance between the target vector and preformed N-1 vectors as (8)

$$d(i)=\left\|\mathbf{X}(i)-\mathbf{X}(N)\right\|, i=1,2,......N-1 \tag{8}$$

2. Select p distances that are most close to the vector $\mathbf{X}(N)$, then select p vectors and value D_r corresponding to the p distances.

$$\mathbf{X}_r^n=[x(t_r),x(t_r+\tau),\cdots x(t_r+(m-1)\tau)]^T,\ r=1,2,\cdots p \tag{9}$$

$$D_r = x(t_r + T)\,,\ r = 1,2,\cdots p \tag{10}$$

For one-step prediction, $T = t_r + m\tau$.

3. $\mathbf{X}_r^n, r = 1\cdots p$ are taken as the input vectors of the SVM and D_r, $r = 1,2,\cdots p$ is taken as the output values gained from training of the SVM.
4. **X**(*N*) is used as an input vectors for the SVM, then the prediction value *x* (*n*+*T*) can be obtained.
5. Iterate step 1-4 till obtaining all prediction values.

Because the samples most close to the target vector are used for training the SVM at every time of iteration, the quadratic programming problem is then simplified. If combined with the SMO or LS-SVM methods, the SVM training speed will increase greatly. The disadvantage of the local method is that for the new input vectors **X**(*N*+1), it must select the local subset and train the SVM over again. That is to say, it just constructed a local dynamic system model. Therefore this method is not suitable for regression, although it has better performances than global SVM.

4 Simulations and Application

4.1 Local ε -SVM Prediction for Logistic Sequence

To verify the prediction performance, a 1000 point Logistic signal was generated from equation (11) for modeling and predicting.

$$x(n+1) = ax(n)(1 - x(n)) \tag{11}$$

Initial value was 0.8 and *a*=4. The first 800 points were used for training the SVM, and the following 200 points were used for testing. According to the Takens's theorem, the embedding dimension was set *m*=3 and the delay time was set τ =1 to reconstruct the phase space. The hidden center of BP neural network and RBF neural network were both set to 10 and the training precision was 0.005. The parameters were set as C=1000 and ε =0.01 in training the ε -SVM. In the local SVM training, close points was selected as *p*=20. The mean square error (MSE) was used here to evaluate the prediction performance as defined in equation (12):

$$e_{MSE} = \frac{1}{N}\sum_{k=1}^{N}\left|x(k) - \hat{x}(k)\right|^2 \tag{12}$$

Table 1 lists the prediction MSE of four learning machines. It's obvious that the MSE of SVM is less than the neural network. In fact, there is no way to compare the MSE between the neural networks and the SVM, because they have different training parameters. But if selecting smaller training precision, the prediction MSE of BP network and the RBF network increase inversely. It shows that the SVM, which is based on the structure risk minimization, performs better than the neural network which is based on the empirical risk minimization.

On the Legend computer which has Pentium(R) 4 CPU 1.80GHz, 256M RAM memory, with the platform of matlab7.0, using the function "QUADPROG" which embed in the matlab7.0 to solve the global ε -SVM directly, the training time was

Table 1. MSE of four learning machine

	BP	RBF	Global ε -SVM	Local -SVM
MSE	3.0882 e-4	3.1979 e-4	3.6377 e-5	3.0000 e-5
Time(s)	-	-	24.13	11.97

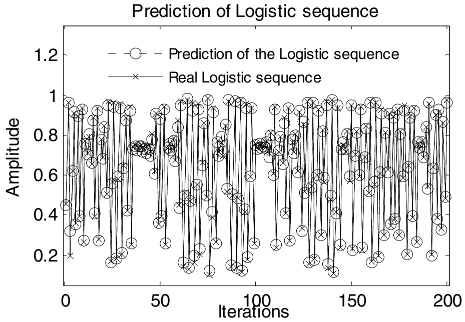


Fig. 2. Prediction of logistic sequence based on local SVM

more than 24 hours. In the same settings, training the local ε -SVM with the function "QUADPROG" needed 81 seconds. Also applying the SMO algorithm in the LIBSVM toolbox, the times for training global ε -SVM was 24.13 seconds. While for local ε -SVM, training 20 close samples only needed 0.0062 second, the total times for predicting 200 point include the time used in sorting and searching the close points is 11.97 seconds. Therefore we can conclude that local ε -SVM has not only smaller prediction MSE but also fast prediction speed. Figure 2 shows the result of prediction for logistic sequence.

4.2 Local ε -SVM for Real EEG Prediction

To evaluate the performance of the proposed method, EEG signal, taken from Mental Health Center in Shantou University, is analyzed. The EEG signal records were collected from normal person who was closing his/her eyes and kept silent and level-headed. Fourteen electrodes were placed according to the international standard 10-20 system. In this section a segment of these real spontaneous EEG signal with 1000 data points was selected for the purpose. The first 800 points were used for training the SVM and the following 200 points were used for testing. The embedding dimension was selected as m=5 and delay time as τ =1. Then constructed phase space according to the Takens's theorem. The training parameters C=1000 and τ =0.01 were set in both global ε -SVM and local ε -SVM. Close points was selected as p=25.

Table 2 lists the training MSE of two SVM and its iterations times using SMO in the LIBSVM toolbox. Figure 3 shows the prediction result of global and local SVM and their prediction errors are depicted in figure 4. It's easy to see from figure 3 that the prediction MSE of global SVM is 3.1084e-3 and the prediction MSE of local

Table 2. Prediction MSE and Training Iteration Times

Method	Global ε -SVM	Local ε -SVM
MSE	3.1084 e-3	2.2000 e-3
Time(s)	583.1	12.42

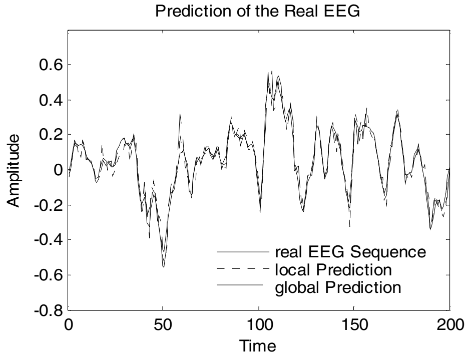


Fig. 3. The real EEG and its prediction

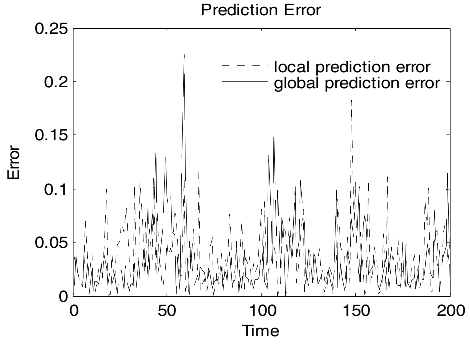


Fig. 4. Comparison of the prediction MSE of the local SVM with the global SVM

SVM is 2.2000e-3. The local SVM improves the prediction precision. Also, in figure 4, there are some sparkles of noise in the global prediction, and the prediction at some peak points is not good. This may be caused by the noise in the EEG, or different dynamic system model. In fact, the local method always selects the close points, that is to say it selects the same dynamic system model with the target vector $\mathbf{X}(N)$ more or less, therefore that its prediction MSE is less. In addition, because of the reducing number of the training samples in local method, the training time at every prediction is cut down. Although it's necessary to train the SVM in every prediction process over again, it can still reduce the total training times. It shows that the time reduces almost 50 times from the table 2.

5 Conclusion

This paper investigates the problem of prediction of EEG signals by using SVM method. A local SVM procedure was proposed to deal with the problem of training global SVM when it comes to a large of training samples. A local SVM based on the local procedures was proposed and combined with the existing training algorithms, such as SMO, to predict EEG signals. The simulation results indicated that the presented local SVM can not only increase the training speed but also effectively reduced the prediction MSE.

Acknowledgements. This research is supported by the Natural Science Foundation of China (Grant No.60571066).

References

1. Lasemidis, L.D., Sackellares, J.C.: Chaos Theory and Eilepsy. The Neuroscientist 2, 118–126 (1996)
2. Lin, X.B., Qiu, T.S.: EEG Signal Analysis and Processing based on Prediction of Epileptic Seizures and Research Progress. Biomedical Engineering Foreign Medica1 Sciences 27, 9–12 (2004) (in Chinese)
3. Takens, F.: Dynamical Systems and Turbulence. Spring Verlag, Berlin (1981)
4. Kaneko, K.: Pattern Dynamic in Spatiotemporal Chaos. Phisica D 34, 1–44 (1989)
5. Leung, H., Haykin, S.: Detection and Estimation using an Adaptive Rational Function Filter. IEEE Trans. Signal Processing 42(11), 3366–3376 (1994)
6. Leung, H.: Nonlinear Clutter Cancellation and Detection using a Memory-based Predictor. IEEE Trans. Aerosp. Electron. System. 32(4), 1249–1256 (1996)
7. Lo, J.T.: Synthetic Approach to Optimal Filtering. IEEE Trans. Neural Network. 5(5), 803–811 (1994)
8. Vapink, V.P.: The Nature of Statistical Learning Theory. Springer, Heidelberg (1995)
9. Smola, A.J., Scholkopf, B.: A Tutorial on Support Vector Regression. Statistics and Computing 14, 199–222 (1998)
10. Burges, C.J.C.: A Tutorial on Support Vector Machines for Pattern Recognition. Data Mining and Knowledge Discovery 2(2), 121–167 (1998)
11. Zhang, J.S., Dang, J.J., Li, H.C.: Spatiotemporal Chaos Sequence Prediction using Local Support Vector Machine. Acra Physica Sinca 56, 67–77 (2007) (in Chinese)
12. Colin, C.: Algorithmic Approaches to Training Support Vector Machines: A Survey. In: Proceedings of ESANN 2000, pp. 27–36. D-Facto Publications, Belgium (2000)
13. Joachims, T.: Making large Scale SVM Learning Practical. In: Advances in Kernel Methods: Upport Vector Machines. MIT Press, Cambridge (1998)
14. Platt, J.C.: Fast Training of Support Vector Machines using Sequential Minimal Optimization. In: Advances in Kernel Methods: Support Vector Learning, pp. 185–208. MIT Press, Cambridge (1999)
15. Keerthi, S., Shevade, S., Bhattcharyya, C., et al.: Improvements to Platt's SMO Algorithm for SVM Classifier Design. Neural Computation 13(3), 637–649 (2001)
16. Suykens, J.A.K., Vandewalle, J.: Least Squares Support Vector Machine Classifiers. Neural Processing Letters 9, 293–300 (1999)

Part IV

Neural Networks as a Soft Computing Technology

Suitability of Using Self-Organizing Neural Networks in Configuring P-System Communications Architectures

Abraham Gutiérrez, Soledad Delgado, and Luis Fernández

Natural Computing Group - Universidad Politécnica de Madrid
Carretera de Valencia Km. 7, Madrid - 28031, Spain
{abraham,sole,setillo}@eui.upm.es

Abstract. Nowadays, it is possible to find out different viable architectures that implements P Systems in a distributed cluster of processors. These proposed architectures have reached a certain compromise between the massively parallelism character of the system and the evolution step times. They are based in the distribution of several membranes in each processor, the use of proxies to control the communication between membranes and mainly, the suitable distribution of the architecture in a balanced tree of processors. For a given P-system and *K* processors, there exists a great volume of possible distributions of membranes over these. The main disadvantage related with these architectures is focused in the selection of the distribution of membranes that minimizes the external communications between them and maximizes the parallelism grade. In this paper, we suggest the use of Self-Organizing Neural Networks (SONN) with growing capability to help in this selection process for a given P-system.

1 Introduction

Possibilities offered by Natural Computation and, specifically P-Systems, for solving NP-problems, have made researchers concentrate their work towards HW and SW implementations of this new computational model. Transition P-Systems were introduced by Păun [1], and were inspired by "*basic features of biological membranes*". One membrane defines a region where there are a series of chemical components (*multisets*) that are able to go through chemical reactions (*evolution rules*) to produce other elements. Inside the region delimited by a membrane can be placed other membranes defining a complex hierarchical structure that can be represented as a tree. Generated products by chemical reactions can remain in the same region or can go to another region crossing a membrane.

A P-System is a computational device which is an abstract representation of a particular membrane structure. Each region is populated by a *multiset* of symbols. These *multisets* are materialized as strings of symbols. In addition, each region is associated with a set of rewriting rules. These rules are applied to the *multisets* (strings of symbols) of certain compartments and, consequently, change the system's configuration. The system configurations are determined by the membrane structure and *multisets* present inside membranes. The rules are applied simultaneously by observing the so-called *maximal parallelism* principle, that is the rules are selected in such a way that only "optimal" output is yielded. When it is not possible to apply any rule, the

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 437–444, 2009.

P-System halts. A designated compartment, called the output compartment, contains the output of the computation, which is equal to the cardinality of the *multiset* contained in it.

In the formal Transition P-Systems model can be distinguished two phases in each evolution step: rules application and communication. Once application rules phase is finished, then it begins communication phase, where those generated *multisets* travel through membranes towards their destination in case it is another region. These systems carry out computations through transitions between two consecutive configurations, what turn them into a computational model with the same capabilities as Turing machines.

Power of this model lies in the fact that the evolution process is massively parallel in application rules phases as well as in communication phase. The challenge for researchers is to achieve hardware and/or software implementations of P systems respecting the massively parallelism in both phases.

Nowadays, it is possible to find out at least three different viable architectures that implements P Systems in a distributed cluster of processors: P2P [2], Hierarchical P2P [3] and Master-Slave [4]. These proposed architectures have reached a certain compromise between the massively parallelism character of the system and evolution step times. In particular, they have focused in the second phase of an evolution step and have obtained good results in the throughput in the external communication among processors and parallelization levels of the system. These architectures are based in the distribution of several membranes in each processor, the use of proxies to control the communication between membranes and mainly, the suitable distribution of the architecture in a balanced tree of processors. These solutions avoid communication collisions, and reduce the number and length for communication among membranes. All this facts allows obtaining a better step evolution time than in others suggested architectures congested quickly by the network collisions when the number of membranes grows. The main disadvantage related with these architectures is the great volume of possible combinations of membrane distributions over a number of processors that can vary from one to the number of membranes. If one processor is used, all of membranes run in the same processor, so external communications is reduced to zero but parallelization level disappears (all internal communications are sequential). On the other side, if each membrane runs in one processor, the better parallelization level is obtained but the increase of external communication produces network congestion and the worst evolution step times. The best solution is the balanced one where internal and external communications remain equilibrated.

In this paper, we suggest the use of Self-Organizing Neural Networks (SONN) with growing capability, based in Fritzke work [5], to help in the search and selection of the balanced distribution for a given P-system, with the purpose of obtaining as a final objective the reduction of the run times of each step of evolution in this P System.

2 P System Communication Architectures

The viable architectures that implements P Systems in a distributed cluster of processors are based on the following:

Membranes distribution: In each processor, *K* membranes are located that will evolve, at worst, sequentially. The value of *K* is determined by the relation between the number of membranes *M* and processors P, where $K \geq 1$. The benefit obtained is that the number of the external communications decreases. The total number of communications splits in two classes: a group of internal communications for pairs of membranes located in the same processor and another group of external communications to interchange information among pairs of membranes located in different processors. Therefore, the number of external communications against the previous model will always be smaller. Moreover, this is an important fact because the run time to carry out the internal communications will be negligible.

For example, the *22* external communications performed by an architecture with a membrane located in each processor (figure 1.a) have been reduced to 10 in the architecture that has located *3* membranes in *4* processors (Figure 1.b).

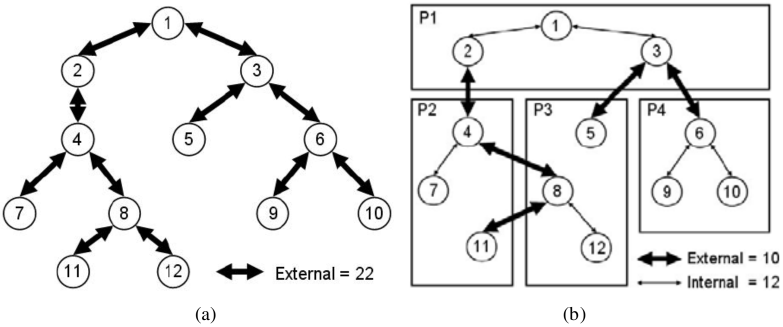


Fig. 1. (a) P system communications. **(b)** Communications with membranes distribution.

Proxy for processor: When a membrane wants to communicate with another one located at a different processor, the first one uses a proxy (programs or device located in the processor that carries out an action in representation of another), instead of doing it directly. Therefore, the communications that use the common line (external communications to the processor) are carried out between proxies, not between membranes. This intermediate element located between the bus and the membranes concentrates the information in two stages:

a) *N multisets* of *N* membranes located in a processor that has a common father membrane in another processor, becoming integrated in a single *multiset* that is the one that will be sent.

b) The *S* communication packet of *L* length necessary to communicate between S pairs of membranes located in 2 different processors are reduced to one single packet of *S.L* length.

The benefit of using proxies in the communication among membranes against direct communication is double:

a) Due to the first stage previously described, the amount of information sent is smaller. This is produced by the fact that the *N* packet necessary to communicate *N* membranes with the same father, are transformed into a single packet of the length of a single *multiset.*

b) Due to the second stage, the number of external communications is smaller although packets are bigger. But, considering that the communication protocols penalize the transmission of small packets because to the data encapsulated processes and to the time safety intervals between future transmissions, it is better to send one packet of length equal $S.L$ than S packets of length equal L.

Figure 2.a shows that if proxies are introduced in the processors, then the number of external communications is reduced to 8.

Tree topology of processors: In graph theory it is established that $P - 1$ connections is the minimum number required to interconnect a connected graph of P processors. This restriction imposes on the graph a tree topology. The benefit obtained with the tree topology of processor is that it minimizes the total number of external communications made as the proxies interchange information only with its direct predecessor and its direct successors, and therefore the total number of external communications in each evolution step is $2(P - 1)$.

Figure 2.b shows that external communications are reduced to 6 when a tree topology of processors is used to connect them.

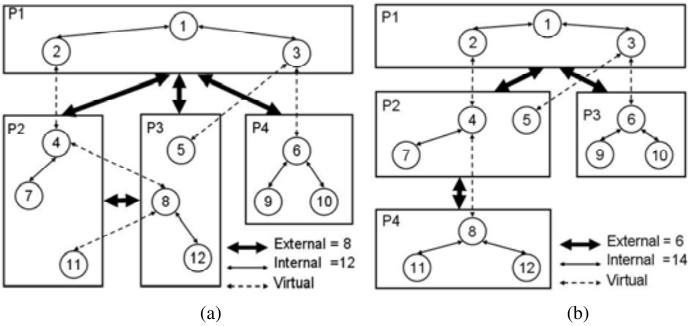


Fig. 2. **(a) Communications with a proxy for** processor. **(b)** Communications using a tree topology.

3 Fritzke's Self-Organizing Neural Networks (SONN)

Self-Organizing Map (SOM) is an artificial neural network model with competitive and unsupervised training. SOM network has two main characteristics: it makes possible obtaining a simplified model of the training data (normally high-dimensional) and it has the capacity to project them on a two dimensional map that shows the existing relations among them. In 1982, Kohonen [5] proposed first model of SOM, where the complete network structure had to be specified in advance and remained static during all the training process. When Kohonen's SOM is used, the choice of inappropriate parameters to define the architecture can degrade the posterior performance of the network. In 1994, B. Fritzke [6] proposed a SOM model called Growing Cell Structure (GCS) where this static structure limitation was eliminated. In addition, the flexibility that offers the possibility of inserting and removing neurons on the output layer of the network during the training phase causes that the GCS network receives better value associated to the topology preserving term, understanding this like the grade that defines the quality of the simplified model that the network represents. There exist other models of SOM networks that usually offer better topology preserving grades than GCS, as GNG[7] and ESOM[8], where there no exist so strict structural organization of the output layer. Nevertheless, this feature causes that they cannot be used directly to generate two-dimensional graphs to show the relations of the input patterns, being necessary using dimension reduction algorithms (like Sammon's projection) to visualize the prototype nodes and their relations. With the purpose of exploiting this characteristic when the input patterns present more than two dimensions we decided to use GCS model.

GCS is a two-layer architecture network (Fig 3). Neurons located at the input layer are fully connected with those in the output one. These connections have associated a weight, w_{ij}, where i identify the input neuron and j the output one. There exist as many input neurons as dimension has the input vectors. Neurons in the output layer have neighbor connections between them presenting a topology formed by groups of basic k-dimensional hyper-tetrahedrons structures. In order to facilitate the visualization of the output layer, in this work a value of k=2 has been used, so output units are connected forming triangle groups.

Every output c unit has an n-dimensional synaptic vector $w_c = (w_{1c}, \ldots, w_{nc})$ associated. This vector can be seen as the position of c in the input vector space. Each time a new input pattern $e = (e1, \ldots, en)$ is processed, only one output neuron is activated, called the *best matching unit (bmu)*, that is the one with the synaptic vector that matches best with the input pattern. Formally:

$$S_{bmu} = \arg\min \left\| e - w_c \right\| . \tag{1}$$

Thereby ‖·‖ denotes the Euclidean vector norm. By this the input vector space is partitioned into a set of regions, each consisting of the locations having a common nearest synaptic vector. This way, the set of all synaptic vectors of the output layer can be seen as a simplified model of the input vector space.

The training phase in GCS network adapts synaptic vectors looking for that each output neuron represents a group of similar input patterns. At the beginning of the training phase the output layer of the network has only three neurons interconnected via neighbor relations (k=2). During the training process a set of input patterns is presented to the network iteratively. In each adaptation step an input pattern is processed, the *bmu* is calculated and its synaptic vector and its topological neighbor's synaptic vectors are modified using equations 1 and 2 respectively (where $\varepsilon_b > \varepsilon_n$).

$$\Delta w_{bmu} = \varepsilon_b \left(e - w_{bmu} \right) . \tag{2}$$

$$\Delta w_C = \varepsilon_n \left(e - w_C \right) \text{ (for all } c \text{ neigbor of } bmu\text{)} . \tag{3}$$

After a fixed number of adaptation steps a new output unit is inserted and is connected to other cells in such a way that the triangular groups of neighbor units are guaranteed. The place where new unit is inserted is determined using two different criteria: "*looking for the unknown probability distribution of the input patterns*" (LUPD) or "*looking for equalize accumulated error measure*" (LEAE) [6]. Periodically superfluous neurons are removed in order to obtain better results when input

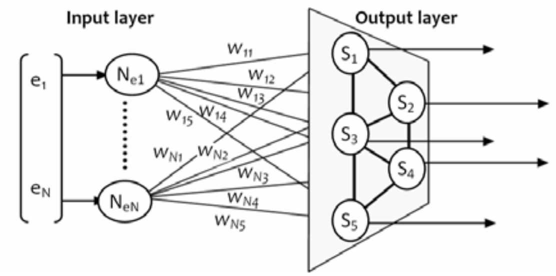


Fig. 3. GCS network topology with *k=2*, N input neurons and 5 output units that exhibit neighbors connections in groups of figures of triangles

space consists of several separate regions of positive probability density. An output neuron can be considered superfluous if it has a synaptic vector in a region with very low probability density (a region without any training pattern). A constant threshold, η, is used to eliminate those neurons with probability density below this value. The removal process ensures the triangular architecture of the output layer, but the output neighbor mesh can results broken in several sub-meshes. In this work the modification of the GCS training algorithm proposed in [9] has been used in order to achieve a better interpretation of the removal parameters.

In a trained network the output layer map can be seen as a projection of the input vector space in a bi-dimensional plane that exhibits the relations of the input patterns. Printing the output layer map data inherent knowledge can be discovered.

4 Experiments and Results

To test the system we have selected thirty P-System models generally used in the literature of P-System. Different membrane distributions between different number of processors have been generated for every P-System, observing how the resulting communications of the distribution affect to the parallelization grade. For each data set associated to a concrete P-System diverse GCS networks have trained with the intention of visualize the output layer and establish the optimal distribution, which will be the one that balances the degrees of external communications and parallelization.

For space reasons in this section only the results of one of the multiple GCS trained networks is showed, in particular the one trained with the P-System of the fig. 1.a, with distributions that uses four processors. First of all, 15 feasible combinations of 12 membranes have been generated in 4 processors that have resulted in 180 bi-dimensional labeled vectors. Each vector maintains the volume of internal and external communications for a concrete membrane-processor-distribution and it has associate a label that identifies these three elements. With this patterns a GCS network has been trained, with LEAE insertion criterion, $\varepsilon_b = 0.06$, $\varepsilon_n = 0.002$, $\mu = 0.006$, and concluding when at least 6 isolated clusters of output units are obtained. After the training phase the output units of the network has been marked with the union of the labels of all those input patterns that fall inside its Voronoi region. Figure 4 shows the scattergram of this GCS network, where the position of each output unit is determined by the two components of its synaptic vector. X-axis coordinates indicates the internal

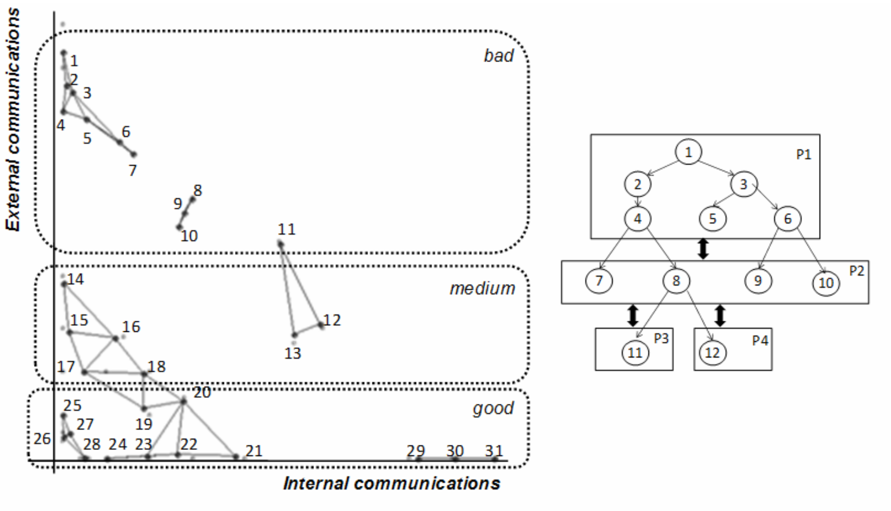


Fig. 4. Left: Scattergram of a GCS network. Points represent neurons and lines between them neighbor connections. Units are grouped in *bad*, *medium* and *good* classes on the basis of the external communication degree. Right: Distribution of membranes C_{13}.

degree of communications and Y-axis the external one. We have grouped the neurons into three classes: bad (from 1 to 11), medium (from 14 to 18) and good (from 19 to 28). Within each of these three groups we have ordered neurons from highest to lowest level of external communication and for those with a similar value, from highest to lowest level of internal communication. Based on this information has been determined that the best distribution is the C13, that contains 1 bad neuron, 4 medium neurons and 7 good neurons. Moreover, this distribution has one of the best ratios of communication (with a volume of 183 for internal communications and 83 for external communications).

5 Conclusions

GCS networks have demonstrated to be a useful tool to P-System in the searching of membrane balanced distributions. Although the example that has been used to document the methodology has a small volume of membranes, the feature of simplified model associated to GCS networks allows working with high volumes of membranes where the distribution possibilities go off.

The analysis of the information of a GCS network could be automated for feeding a system of automatic membrane distribution over processors. In particular, this tool is being adapted to be used in the distributed system of membranes based on microcontrollers exposed in [10][11].

Given the good results obtained in the experiments, as future extensions the possibility of working with vectors of greater dimension is considered, what will allow to fit the search of suitable balanced P-System, for example generating a single vector for each membrane distribution or working with fuzzy values for determining the degree of internal and external communications of a processor. This will require working with new visualizations of the output layer of the GCS network, such as those exposed in [12].

References

1. Păun, G.: Computing with membranes. Journal of Computer and System Sciences, 61 (2000), and Turku Center for Computer Science-TUCS Report No 208 (1998)
2. Tejedor, A., Fernandez, L., Arroyo, F., Bravo, G.: An architecture for attacking the bottleneck communication in P Systems. In: Sugisaka, M., Tanaka, H. (eds.) Proceedings of the 12th Int. Symposium on Artificial Life and Robotics, Beppu, Oita, Japan, pp. 500–505 (2007)
3. Bravo, G., Fernández, L., Arroyo, F., et al.: A hierarchical architecture with parallel communication for implementing P-Systems. In: ITA 2007 Xth Joint International Scientific Events on Informatics, Varna, Bulgaria (2007)
4. Bravo, G., Fernández, L., Arroyo, F., et al.: Master-Slave Distributed Architecture for Membrane Systems Implementation. In: 8th WSEAS Int. Conf. on Evolutionary Computing (EC 2007), Vancouver, Canada (2007)
5. Kohonen, T.: Self-Organized Formation of Topologically Correct Feature Maps. Biological Cybernetics 4359–4369 (1982)
6. Fritzke, B.: Growing Cell Structures – A Self-organizing network for Unsupervised and Supervised learning. Neural Networks 7(1), 1441–1460 (1994)
7. Fritzke, B.: A growing neural gas network learns topologies. Advances in neural information processing systems 7, 625–632 (1995)
8. Deng, D., Kasabov, N.: On-line pattern analysis by evolving self-organising maps. Neurocomputing 51, 87–103 (2003)
9. Delgado, S., Gonzalo, C., Martínez, E., Arquero, A.: Improvement of Self-Organizing Maps with Growing Capability for Goodness Evaluation of Multispectral Training Patterns. In: IEEE International Geoscience and Remote Sensing Symposium, vol. 1, pp. 564–567 (2004)
10. Gutiérrez, A., Fernández, L., Arroyo, F., Alonso, S.: Hardware and Software Architecture for Implementing Membrane Systems: A Case of Study to Transition P Systems. In: Garzon, M.H., Yan, H. (eds.) DNA 2007. LNCS, vol. 4848, pp. 211–220. Springer, Heidelberg (2008)
11. Gutierrez, A., Fernández, L., Arroyo, F., Alonso, S.: Suitability of Using Microcontrollers in Implementing new P System Communications Architectures. In: AROB 2008, XIII International Symposium on Artificial Life and Robotics, Oita, JAPAN, January 31-February 2 (2008)
12. Delgado, S., Gonzalo, C., Martinez, E., Arquero, A.: Visualizing High-Dimensional Input Data with Growing Self-Organizing Maps. In: Sandoval, F., Prieto, A.G., Cabestany, J., Graña, M. (eds.) IWANN 2007. LNCS, vol. 4507, pp. 580–587. Springer, Heidelberg (2007)

Short Term Load Forecasting (STLF) Using Artificial Neural Network Based Multiple Lags of Time Series

Mohd Hafez Hilmi Harun, Muhammad Murtadha Othman*, and Ismail Musirin

Faculty of Electrical Engineering, Universiti Teknologi MARA, 40450 Shah Alam,
Selangor, Malaysia
mamat505my@yahoo.com

Abstract. This paper presents the artificial neural network (ANN) that used to perform the short-term load forecasting (STLF). The input data of ANN is comprises of multiple lags of hourly peak load. Hence, imperative information regarding to the movement patterns of a time series can be obtained based on the multiple time lags of chronological hourly peak load. This may assist towards the improvement of ANN in forecasting the hourly peak loads. The Levenberg-Marquardt optimization technique is used as a back propagation algorithm for the ANN. The Malaysian hourly peak loads are used as a case study in the estimation of STLF using ANN. The results have shown that the proposed technique is robust in forecasting the future hourly peak loads with less error.

1 Introduction

Load forecasting has always been an essential task for the electric utilities in which it may assist to an effective operational planning and security assessment of a power system. This is important to ensure that the electric utilities are operating in an economic, reliable and uninterrupted service to the customers [1]. With the advent of deregulation in electric utilities, load forecasting becomes even more important especially to the system operators and market participants whereby this may assist towards organizing appropriate strategies of risk management and competitive energy trading [1] and [2].

In this paper, ANN is used to perform the short-term load forecasting (STLF) for the next 24 hour. The ANN is an outstanding and promising approach for STLF compared to other methods such as the autoregressive-moving average (ARMA) and regression approach. The main strength of ANN is on its ability to model complex and non linear relationship of the network by training it with the historical data of hourly peak loads. The input data of ANN comprises of multiple lags of hourly peak load. The Malaysian hourly peak loads are used as a case study in the assessment of STLF using ANN model incorporating with the Levenberg-Marquardt (LM) back propagation algorithm. The performances of ANN model in forecasting is investigated based on the accurateness of forecasted hourly peak loads.

* Corresponding author.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 445–452, 2009.

2 Multiple Time Lags of Chronological Variables

Improving the input variables of artificial neural network (ANN) plays an imperative role in order to produce accurate results of forecasting. Equation (**1**) is used to obtain an improved input data of ANN that takes into account the multiple time lags of chronological hourly peak load, X_{t-k}. By considering the multiple time lags of chronological variables in equation (**1**), this may assist the ANN to easily recognize the movement pattern of each variable at every time interval [3]. Hence, this may alleviate the performance of ANN in providing better results of forecasting.

$$Z_{k,t} = X_t - X_{t-k} \tag{1}$$

where,

X_t : time series of hourly peak load.
t : time interval.
k : lagging time interval which is 1, 2, 3,…., K.
K : total number of lagging time interval.

By using equation (**1**), the input data of ANN is in a *k-by-t* matrix form. In this case study, each column of matrix is used by the ANN to forecast one variable at the next 24 hour. Walczak et al. [3] explained that the total number of lagging time interval, K, is initially specified equivalent to the time interval of forecasted variable. In conjunction to the above mentioned explanations, the arrangement of ANN input data with respect to the observed data or targeted data is depicted in Figure 1. For instance, the first column of input data, $Z_{k,t}$, is used by the ANN to forecast the targeted variable of X_{48} that is the peak load at the next 24 hour.

$$\text{Training Data } (Z_{k,t}) = \begin{pmatrix} X_{24}-X_{23} & X_{25}-X_{24} & \ldots\ldots & X_{6999}-X_{6998} & X_{7000}-X_{6999} \\ X_{24}-X_{22} & X_{25}-X_{23} & \ldots\ldots & X_{6999}-X_{6997} & X_{7000}-X_{6998} \\ X_{24}-X_{21} & X_{25}-X_{22} & \ldots\ldots & X_{6999}-X_{6996} & X_{7000}-X_{6997} \\ X_{24}-X_{20} & X_{25}-X_{19} & \ldots\ldots & X_{6999}-X_{6995} & X_{7000}-X_{6996} \\ . & . & & . & . \\ . & . & & . & . \\ X_{24}-X_{2} & X_{25}-X_{3} & \ldots\ldots & X_{6999}-X_{6977} & X_{7000}-X_{6978} \\ X_{24}-X_{1} & X_{25}-X_{2} & \ldots\ldots & X_{6999}-X_{6976} & X_{7000}-X_{6977} \end{pmatrix}$$

$$\text{Target Data} = (X_{48} \quad X_{49} \quad X_{7023} \quad X_{7024})$$

Fig. 1. The arrangement of input data and targeted data for the ANN training procedure

3 Implementation of Artificial Neural Network

In this paper, the ANN model consists of one input layer, two hidden layers and one output layer .The output layer of ANN is consisting of one neuron that is to provide the result of hourly peak load at the next 24 hour. The linear activation function is used as a neuron in the output layer of ANN. In particular, the optimization process is performed by using the Levenberg-Marquardt (LM) technique in which it is adapted as the back propagation algorithm of ANN. The LM is widely accepted as an efficient algorithm in terms of accuracy and it also give a good compromise between the speed of the Newton method and the stability of the steepest descent method [4].

The logistic tangent function is used as the neurons for each hidden layer since it process the input data that is in the range of 0 to 1 per unit (p.u.). The per unit value is obtained by the actual value of hourly peak load divided by the maximum value of hourly peak load.

In order to avoid over-training of ANN therefore, cross-validation of ANN training procedure is performed to ensure accuracy of the results. The input data of ANN is divided into three sets whereby 80%, 10% and 10% of the ANN input data are used for training, validation and testing of ANN, respectively. In the training algorithm of ANN, the network minimizes the error between the output and desired output by adjusting the weight and biases. The error minimization process is repeated until it converges to a predefined small value of error. Then, the training procedure of ANN is terminated once the minimum error becomes constant.

The number of hidden layer is selected based on the fact that one hidden layer is enough to approximate any function, although two hidden layers may useful in some circumstances [5]. The choices of selecting the number of neurons in the hidden layers are made through cross validation of ANN training procedure which gives to a minimum root mean square (rms) error of ANN output. Then, the testing procedure of ANN is performed in order to obtain the results of load forecasting for the next 24 hour.

4 Results and Discussion

The Malaysian hourly peak loads in the year 2002 are used as a case study in the assessment of STLF using the ANN. The hourly peak loads with the total time interval of 8736 hour is shown in Figure 2 and it is divided into three sets for the training, validation and testing procedures of ANN.

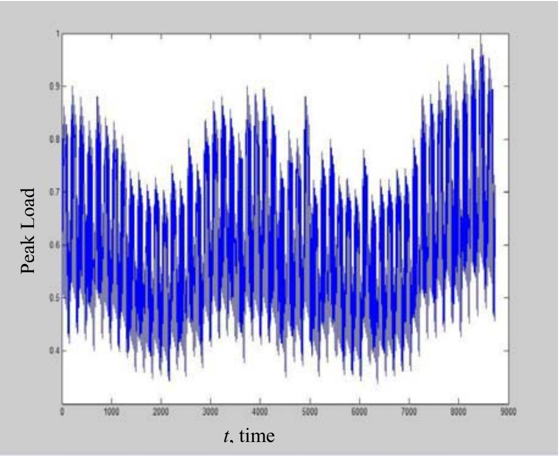


Fig. 2. The Malaysian hourly peak loads in the year 2002

The total number of lagging time interval should be initially specified to K = 24 hour since the future hourly peak load is forecasted at the next 24 hour. The input data with K = 24 hour is used by the ANN which consists of 14 and 8 neurons in first and second hidden layers, respectively. The input data with K= 48 hour is used by the ANN which consists of 19 and 16 neurons in the first and second hidden layers, respectively. Finally, the input data with K = 72 hour is used by the ANN that consists of 13 and 7 neurons in first and second hidden layers, respectively. The numbers of neurons in the first and second hidden layers are selected based on the minimum amount of rms output error obtained through the cross-validation of ANN as shown in Table1 and Table 2, respectively.

Table 3 shows that the training and validation processes of ANN provide less output error for K = 72 hour compared to K = 48 and K = 24 hours. The output results are obtained by considering different ANN architectures specified for K = 24 hour, K = 48 hour and K = 72. It is observed that the total number of lagging time interval, K, significantly affect the performance of ANN in forecasting. It is obvious that the input data of K = 72 hour improves the ANN training and validation processes compared to the input data with K = 24 hour and K = 48 hour.

Table 1. RMS error of ANN output subsequent to the increased number of neurons in first hidden layer

Lag 24			Lag 48			Lag 72		
Number of Neurons	Training	Validation	Number of Neurons	Training	Validation	Number of Neurons	Training	Validation
	RMS Output Error			RMS Output Error			RMS Output Error	
11	0.0028	0.004	15	0.0012	0.0018	8	0.0006	0.001269
12	0.0028	0.0031	16	0.0012	0.0017	9	0.000583	0.001376
13	0.0022	0.0032	17	0.0012	0.0017	10	0.00027	0.000705
14	**0.0019**	**0.0026**	18	0.0012	0.0019	11	0.000239	0.000746
15	0.0018	0.0028	**19**	**0.0011**	**0.0017**	12	0.000271	0.000705
16	0.0017	0.0028	20	0.0012	0.0018	**13**	**0.00023**	**0.00055**
17	0.0019	0.0027	21	0.0011	0.0018	14	0.00028	0.000783
18	0.0017	0.0027	22	0.0011	0.0024	15	0.000334	0.001072

Table 2. RMS error of ANN output subsequent to the increased number of neurons in second hidden layer

Lag 24			Lag 48			Lag 72		
Number of Neurons	Training	Validation	Number of Neurons	Training	Validation	Number of Neurons	Training	Validation
	RMS Output Error			RMS Output Error			RMS Output Error	
5	0.0017	0.0024	12	0.0011	0.0016	4	0.000217	0.000384
6	0.0016	0.0026	13	0.0012	0.0018	5	0.000222	0.000375
7	0.0016	0.0028	14	0.0004	0.0008	6	0.000212	0.000381
8	**0.0015**	**0.0023**	15	0.0015	0.0025	**7**	**0.0002**	**0.00038**
9	0.0015	0.0025	**16**	**0.0004**	**0.0007**	8	0.000225	0.000472
10	0.0017	0.0027	17	0.0017	0.0018	9	0.000286	0.000682

Table 3. Results of ANN considering K = 24 hour, K=48 hour and K= 72

	ANN input data with multiple time lags of K = 24 hour	ANN input data with multiple time lags of K = 48 hour	ANN input data with multiple time lags of K = 72 hour
Training sets	7000		
Validation sets	1016		
Testing Sets	720		
Number of input neurons	23	47	71
Number of neurons in 1st hidden layer	14	19	13
Number of neurons in 2nd hidden layer	8	16	7
Number of output	1		
Learning rate	0.001	0.01	0.1
Momentum	0.095	0.87	0.999
Training function	Levenberg-Marquardt		
Training rms error	0.0025	0.00035794	0.000201094
Validation rms error	0.0024	0.00072269	0.000381081

Figures 3, 4 and 5 represent the result of hourly peak loads forecasted consecutively for the next 24 hour. It is obtained during the testing procedure of ANN that takes into account the input data with K = 24 hour, K = 48 hour and K =72 hour, respectively. The red and black lines represent as the output result of ANN and targeted output of ANN, respectively. Figure 3 represents the output results of ANN based on the input data with K = 24 hour. The multiple time lags should be initially specified to K = 24 hour so that it is equivalent to the 24 hour time interval of forecasted hourly peak load. It is obvious that the multiple time lags of chronological hourly peak load with K = 24 hour does not improve the performance of ANN in forecasting. Whereby, large error can be observed via comparison between the hourly peak loads forecasted by the ANN and the targeted hourly peak loads. Comparative study between the forecasted and actual hourly peak loads have shown that the input data with K= 72 hour alleviate the performance of ANN to accurately forecast the future hourly peak loads at the next 24 hour with less error. This consensus was supported by the comparison between the results of forecasted hourly peak loads depicted in Figures 3, 4 and 5. The comparison results elucidate that the ANN input data with K = 72 hour improves the ANN performance which provide to a better result of STLF as compared to the result provided by the ANN considering the input data with K = 24 hour and K = 48 hour.

The ANN testing procedure with three multiple time lags of chronological hourly peak load are further investigated by referring to the mean absolute percentage error (MAPE) of output results. The formulation of MAPE is given in equation (**2**).

$$MAPE = \frac{1}{N}\sum_{t=1}^{N}\frac{Y_t - X_t}{X_t} \tag{2}$$

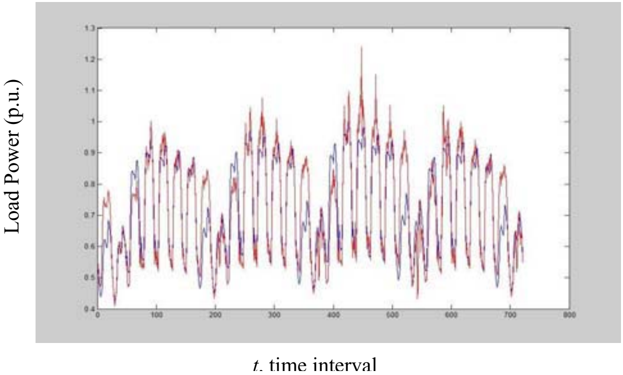


Fig. 3. Comparison between the forecasted hourly peak loads and the actual values based on $K = 24$ hour

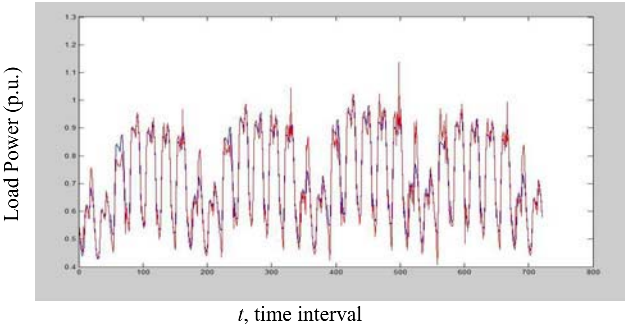


Fig. 4. Comparison between the forecasted hourly peak loads and the actual values based on $K = 48$ hour

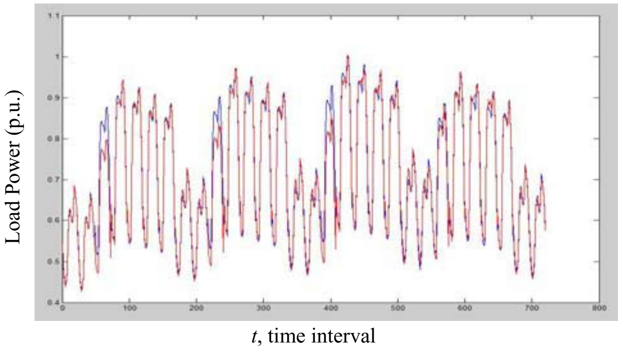


Fig. 5. Comparison between the forecasted hourly peak loads and the actual values based on $K = 72$ hour

where,

Y_t : forecasted hourly peak load at time interval t.
N : total number of time interval.

In the testing procedure of ANN, the MAPE of forecasted hourly peak loads for each day is determined based on the three cases of ANN input data that is K = 24 hour, K = 48 and K = 72 hour. The results of MAPE for each case are shown in Table 4. The ANN input data with K = 24 hour, K = 48 hour and K=72 hour causes the ANN to provide minimum MAPE values of 1.74% at day two , 0.92% at day two, and 0.50% at day two, respectively. This shows that the input data with K = 24 hour does not improve the ANN performance which provide large error in forecasting the future hourly peak loads. This consequence is similar to a case whereby ANN with K = 24 hour provides a large MAPE value of 19.71% at day 15 as compared to the 8.99% at day 22 and 9.14% at day 17 for the ANNs with K = 48 hour and K = 72 hour, respectively. On the other hand, the average MAPE values of 7.45%, 4% and 1.95% are obtained based on the input data of ANN considering K = 24, K = 48 and K = 72 hours, respectively. It is obvious that the input data with K = 72 hour increases the performance of ANN in forecasting the future hourly peaks loads with less error as compared to the ANN considering the input data with K = 24 hour and K= 48 hour.

The results showing that it is important to accurately specify the total number of lagging time interval, K, for the input data in which this may significantly affect the performance of ANN in obtaining the STLF. The advantage of utilizing the proposed approach is that the ANN able to perform STLF by considering only the input data of univariate time series that is the chronological hourly peak load. Furthermore, other factors such as the temperature, weather and energy market can also be included as the input data to improve the performance of ANN in STLF. However, this may yields to a large size of ANN input data. Indeed, it can be reduced by extracting or

Table 4. MAPE of the forecasted hourly peak loads considering K=24 hour, K = 48 hour and K=72 for ANN

Day	K = 24 hour	K = 48 hour	K = 72 hour	Day	K = 24 hour	K = 48 hour	K = 72 hour
	MAPE (%)	MAPE (%)	MAPE (%)		MAPE (%)	MAPE (%)	MAPE (%)
1	14.73	4.97	0.68	**16**	5.59	2.75	1.45
2	1.74	0.92	0.50	**17**	12.33	6.28	9.14
3	9.17	7.85	8.50	**18**	5.18	2.57	1.17
4	3.98	1.75	0.92	**19**	7.43	3.01	1.37
5	3.83	1.86	0.84	**20**	6.26	3.60	1.24
6	2.21	1.99	0.86	**21**	4.55	5.53	1.18
7	1.79	3.33	0.75	**22**	22.23	8.99	1.96
8	17.71	6.61	1.30	**23**	7.48	3.71	1.8
9	4.08	1.98	1.12	**24**	8.26	6.79	2.92
10	10.2	5.81	7.98	**25**	6.93	3.18	1.08
11	4.28	1.99	0.91	**26**	4.37	2.53	1.4
12	5.38	2.35	1.05	**27**	2.65	2.26	1.35
13	3.82	2.75	0.97	**28**	1.94	3.62	0.82
14	2.59	4.23	0.96	**29**	18.39	6.94	1.41
15	19.71	7.64	1.62	**30**	4.61	2.24	1.24

selecting only the significant input features and this may further improve the performance of ANN in STLF.

5 Conclusion

The application of artificial neural network (ANN) in performing the short-term load forecasting (STLF) has been presented. The Malaysian hourly peak loads have been used as a case study in the assessment of STLF using ANN. The univariate time series of hourly peak loads is converted to a multiple time lags of time series and it is then used as an input data of ANN. The transformation that improves the input data may alleviate the performance of ANN in forecasting the hourly peak loads. The Levenberg-Marquardt technique is used to optimize the weight of ANN network interconnections. The results have shown that the proposed method is able to forecast the Malaysian hourly peak load for the next 24 hour with less error.

References

1. Methaprayoon, K.: Neural Network-Based Short Term Load Forecasting for Unit Commitment Scheduling. M.S. thesis, The University of Texas at Arlington, USA (2003)
2. Sisworahardjo, N.S., El-Keib, A.A., Choi, J., Valenzuela, J., Brooks, R., El-Agtal, I.: A Stochastic Load Model for an Electricity Market. Electric Power Systems Research 76, 500–508 (2006)
3. Walczak, S.: An Empirical Analysis of Data Requirement for Financial Forecasting with Neural Networks. Journal of Management Information System 17, 203–222 (2001)
4. Wilamowski, B.M., Kaynak, O., Iplikci, S., Efe, M.O.: An Algorithm for Fast Convergence in Training Neural Networks. In: International Joint Conference on Neural Network, vol. 3, pp. 1778–1782 (2001)
5. El-Desouky, A.A., El-Kateb, M.M.: Hybrid Adaptive Techniques for Electric-Load Forecast Using ANN and ARIMA. In: IEEE Proceedings of Generation, Transmission and Distribution, vol. 147, pp. 213–217 (2000)

Neural Network Regression for LHF Process Optimization

Miroslaw Kordos

Department of Material Engineering and Metallurgy,
The Silesian University of Technology,
Krasinskiego 8, 40-019 Katowice, Poland
miroslaw.kordos@polsl.pl

Abstract. We present a system for regression using MLP neural networks with hyperbolic tangent functions in the input, hidden and output layer. The activation functions in the input and output layer are adjusted during the network training to fit better the distribution of the underlying data, while the network weights are trained to fit desired input-output mapping. A non-gradient variable step size training algorithm is used since it proved effective for that kind of problems. Finally we present a practical implementation, the system found in the optimization of metallurgical processes.

1 Introduction

1.1 Data Distribution

The standard approach to data regression using MLP networks is to apply a 3-layer MLP network with a linear input, logistic sigmoid or hyperbolic tangent (*tanh*) hidden and linear output units [1][2]. A good practice is to standardize the data before the training, e.g. according to the following formulae:

$$x_s = \frac{x - \overline{x}}{\delta} \qquad \delta = \sqrt{\frac{\sum_{i=1}^{k} (\overline{x} - x)^2}{k}} \tag{1.1}$$

to make particular inputs influence independent of their physical range. It may be also be beneficial to remove the outliers from the training set. Moreover, a model with higher sensibility in the intervals with more dense data may be preferred. To address the problem, for example, the data can be transferred according to hyperbolic tangent (Fig. 1.): *y=(1-exp(-βx))/(1+exp(-βx)).* The other advantage of such a transformation is the automatic reduction of the outliers' influence on the model. We do not consider the outliers as erroneous values and thus do not reject them [3], but rather reduce their influence on the final model. The neural network learns the optimal parameters of the transfer function during the training.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 453–460, 2009.

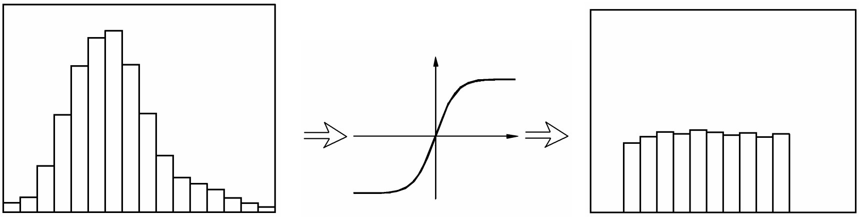


Fig. 1. The idea of transforming data from a Gaussian-like distribution to uniform distribution

1.2 LHF Process

The system (which is shortly outlined in the next chapter) is being implemented in one of the Steelworks in Europe. The metallurgical process consists of three stages. At the first stage an electric arc furnace (EAF) heats steel scrap with an electric arc. Once the temperature and chemistry are roughly correct, the steel is tapped out into a preheated ladle arc furnace (LHF) where various chemical elements (C, Si, Mn, P, …) are precisely added to obtained the desired, final properties of a given kind of steel. In the third step, the steel is formed into the shapes demanded by customers.

The regression task refers to the LHF stage, which is the most difficult and most costly part of the process. Hundreds of variables are measured, including chemical analysis, temperatures, times and others. In the real system, the first feature selection is performed by an expert and then by either forward feature selection with beam search or a feature ranking [4]. For the purpose of the simplicity of this paper we preselected here 26 features and do not discuss the feature selection here.

The task is to predict the quantity of particular elements that must be added to the steel to obtain the desired chemical composition and thus build an expert system that will suggest the proper quantities to the LHF operator. The human operator adds the elements based on combination of the measured quantities and his experience. However, the problem is that there are such elements, which once added in excessive amount can never be removed from the steel. To avoid wasting all the steel in the ladle in this way, the operator adds the elements iteratively, then the next chemical analysis is performed and then the remaining elements are added. That can be repeated several times. There are usually no simple linear relations between the amount of the element added to steel and the amount present in the final product. Some portions the elements get burned or remain on the surface of the steel, other react with each other. In addition, all that depends on time, temperature and other factors. By a proper prediction we avoid excessive number of iterations. That minimizes the time of the LHF process and that in turn reduces production costs [5-9]. The number of the outputs (number of normalized elements) and the required accuracy differs between various steel sorts, being usually between 5 and 10 and the required accuracy is at the level of 1-10%.

2 Method

2.1 System

The system consists of four modules, as shown in Fig. 2. If data from many LHF processes is available for a given sort of steel, we build a separate model for it.

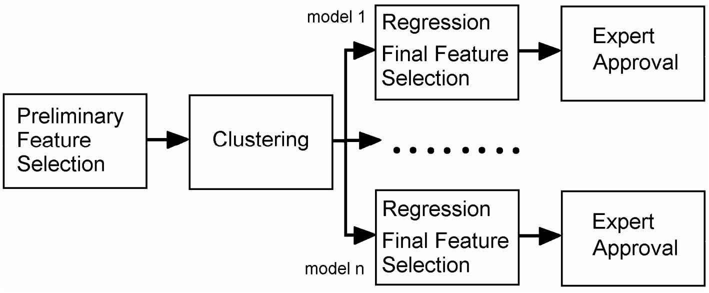

Preliminary Feature Selection
Clustering
model 1
Regression Final Feature Selection
Expert Approval
Regression Final Feature Selection
model n
Expert Approval

Fig. 2. The system for predictions of the amount of elements used in the LHF process

However, in many cases there are only data from a few production cycles of a rarely produces sorts of steel. That does not allow for building a reliable model. Thus, the rarely sort are joined together and then using k-means weighted clustering (weights proportional to the input-output correlation on the whole data [10] Euclidean distance is used) they are grouped into bigger datasets, for which the regression models are built. If it happens that the prediction results for a given cluster are worse than for a whole dataset, then the model built for the whole dataset is used instead of a given cluster model.

In this paper we concentrate on the neural network based regression block. In the real system we use forward feature selection with beam search o feature rankings [4]. Comparing to the 26 preselected features, the selection only slightly improves system generalization, but significantly improves the model readability and its understanding by experts, thus increasing the chance that the expert will approve that model. For example if the expert does not understand why some feature combination influences the result, the feature space must be modified even if the prediction results are satisfactory. Additionally, the MLP network by its nature performs a kind of feature selection by setting the values of the weights that connect inputs from the non-meaningful variables to very small values. Principal Component Analysis is not a good choice here; first the consecutive eignevalues decrease very slowly for this data and second it makes the results very hard to interpret by an expert that must approve the model.)

2.2 Neural Network Architecture

The neural network we use for regression is a 3-layer MLP network with hyperbolic tangent activation functions in input, hidden and output layer. The adaptable parameters are the weights and biases of the hidden and output layer neurons and the slope of *tanh* functions in input and output layer.

The purpose of using *tanh* activation functions in the input layer neurons is to transform the data distribution, as discussed in the introduction. The purpose of using *tanh* activation function in the output layer neurons is to reduce the influence of outliers on the model outcome. Thus, the outliers can be retained in the training set; they can carry some useful information on rare data points. On contrary, with standard MLP network, the best approach would be rather to eliminate the outliers.

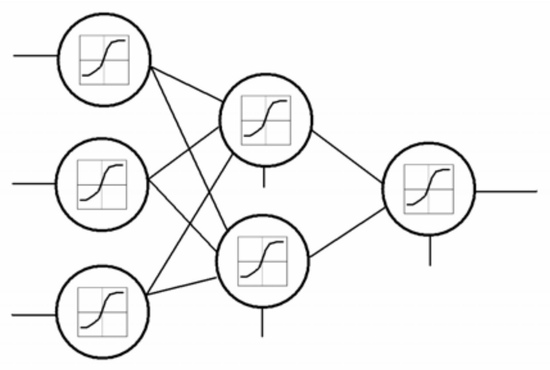


Fig. 3. Architecture of the MLP neural network

The following network parameters are tuned in the system:

- Input layer: β (slope of *tanh* function) or the power in the power function
- Hidden layer: weight and biases
- Output layer: weight, biases and β (slope of tanh function) or the power in the power function

The slope of the activation function of the hidden layer neuron is fixed (β=1) to limit the number of parameters and keep the model simple as for now, however we consider adjusting it in the future work.

2.3 Training Algorithm

For the network training we use a variable search step method (VSS) [11], because it has efficiency comparable to Levenberg-Marquardt algorithm and at the same time it is well suited for big networks due to low memory requirements.

The basic VSS algorithm is very simple and is outlined in the following pseudo code:

```
for i=1 to NumberOfEpochs do
     for j=1 to NumberOfParameters do
          find dwj that minimizes E(i,wj+dwj);
          wj ← wj+dwj;
    end
    if E < Emin
        break;
    end
end
```

Parameters include weight, biases and here we extended the original method on slopes β also. E_{min} is the error (MSE) value at which the training stops; selecting the stopping criterion is not specific to VSS and can be done in the same way as for other MLP training algorithms.

Any line search minimization method can be used to find the optimal *dw*, and the mean-square error (MSE) or any other error measure may be used as the optimization criterion. However, to increase the computational efficiency of VSS algorithm special methods to compute *dw* and *E*(*e,w*+*dw*) maybe used, as outlined below.

Because only one weight (or in general one parameter) is changed at a time the input signals do not need to be propagated through the entire network to calculate the error. Propagation through the fragment of the network in which the signals may change as a result of the weight update is sufficient. The remaining signals incoming to all neurons of hidden and output layers are remembered for each training vector in an array called the "signal table". After a single weight is changed only the appropriate entries in the signal table are updated. The MSE error of each output neuron is also remembered and do not need to be recalculated again if a weight of another output neuron is changed.

The dimension of the signal table is $N_V(N_o+N_h)$, where N_V is the number of vectors in the training set and N_h and N_o are the numbers of hidden and output neurons, respectively. For example, for a network with 30 neurons and 10,000 training vectors, storing variables in 8 bytes (double type) the signal table needs only 2.3 MB of memory, that is two or more orders of magnitude less than the memory requirements for the Levenberg-Marquardt (LM) algorithm, and also less than the requirements of Scaled Conjugate Gradient (SCG) algorithm

The search algorithm should take advantage of the MLP error surface properties. The steepness of the error surface in different directions varies by orders of magnitude, and the ravines in which the MLP learning trajectories lay are usually curved, slowly changing their directions [12][13]. Therefore one can expect that an optimal change of weight value *dw* for the same weight in two successive training cycles will not differ much, while *dw* for different weights in the same training cycle may have values that differ on orders of magnitude. In each training cycle *i* the first guess of *dw*(*w,i*) for a given weight *w* might be the value *dw*(*w,i*–1) of the weight change in the previous training cycle.

A more efficient, however also more complex line search method is described in our last paper [11].

3 Experimental Results

The dataset on which the experiments were performed is available from *http://www.kordos.com/datasets/steel26.zip*. Because the data comes from a real steelworks, it is confidential and could not be released as it is. Thus, the 26 input variable names were replaced by x1, … x26, all other inputs were removed, only four most common elements (C, Si, Mn, P) were left and the data was standardized (zero mean and unit standard deviation).

We tried different number of hidden units in the neural network. The best results were obtained for the number between 25 and 35. Thus, we finally used 28 hidden units for predicting the quantities of each element.

The experiments with MLP neural network were performed with software created by us in Delphi. (*http://www.kordos.com/mknn.html*)

The experiments with Support Vector Regression [14] were performed in the *Statistica* software, using RBF kernels with γ=0.056 and Regression Type 2 (that were the parameters giving the best results). Optimal *C* and *nu* parameters were determined during the training individually for each element. SVR was chosen for comparison because it is an effective method, which even frequently performs better than MLP. However, on this data it did not, except for Silicon.

10 runs of 10-fold crossvalidation were performed and the average results on the test part of the dataset are reported in Table 1 and Table 2. The MSE errors are always reported on the original output data (not on the data transformed by the *tanh* function). The standard deviations were usually of the order of 0.002-0.004 for between-crossvalidation and 0.01-0.03 for within-crossvalidation runs.

The training time of the network was of the order of tens of seconds up to single minutes, depending on the dataset size. However, the time of building a full model (for one sort of steel or for one cluster), including feature selection and selection of optimal neural network architecture (which can be different for each feature subset) can even be of the order of hours on a single processor. That is acceptable in our application, since the model must be built only once. The system must only predict the data on the fly and the prediction is instantaneous.

Table 1. Mean Squared Error of predicting the amount of elements to be added in the LHF process. Prediction performed on the whole dataset. ("preproc." means that the input and output data was preprocessed prior to the training by a constant tanh function, selected manually to give the better transformation to a uniform data distribution.)

	MLP+ATF 26-28-1	MLP 26-28-1	SVM (RBF, γ=0.056)	MLP 26-28-1 + preproc.	SVM (RBF, γ=0.056) + prep.
C	0.0349	0.0409	0.0595	0.0362	0.0498
Si	0.0728	0.0945	0.0650	0.0764	0.0650
Mn	0.1055	0.1189	0.1635	0.1108	0.1543
P	0.1342	0.1482	0.1849	0.1402	0.1422
average	0.0869	0.1006	0.1182	0.0909	0.1028

Table 2. Mean Squared Error of predicting the amount of elements to be added in the LHF process. Prediction performed on 5 clusters, as described in chapter 2.1

	MLP+ATF 26-28-1	MLP 26-28-1	SVM (RBF, γ=0.056)	MLP 26-28-1 + preproc.	SVM (RBF, γ=0.056) + prep.
C	0.0315	0.0377	0.0545	0.0341	0.0475
Si	0.0692	0.0802	0.0640	0.0721	0.0764
Mn	0.0978	0.1107	0.1509	0.1049	0.1470
P	0.1256	0.1405	0.1723	0.1346	0.1362
average	0.0810	0.0923	0.1104	0.0864	0.1018

4 Conclusions

A system for predicting the quantities of additives in the steel production process was presented. The pipeline consists of several blocks. The paper concentrated on the neural network based regression block. We proposed that the input and output variable transformation from a Gaussian-like to uniform-like distribution be performed during the neural network training, incorporated in the VSS training method we use. That brings better results than preprocessing the data before the training, since it allow adjusting the distribution individually for each feature, without a strong a priori assumption that the optimal distribution is always uniform. As it can be seen from the tables, MLP network with adaptable transfer function provides the best results, that probably could be further improved if the selection of the activation functions in the hidden layer be performed during the training [15] (we are planning to include this in our future work).

The goal of the system is to shorten the LHF process time on average by at least 5%, thus reducing the cost of a single melting and consequently to allow producing more steel monthly.

To make the system easier to understand by experts, what is crucial in our application (the experts don't want to allow for a system they do not understand) we are currently working on one more block of the system, located between Regression and Expert Approval: Rule Extraction from Neural Network, using an adaptation of one of Setiono's methods [16].

Acknowledgements

The project was partially sponsored by the grant No 6ZR9 2006 C/06742 from the Polish Ministry of Education and Science (MNiSZW).

References

1. Dasgupta, B., Schnitger, G.: The Power of Approximating: A Comparison of Activation Functions. In: Advances in Neural Information Processing Systems, vol. 5, pp. 615–622. Morgan Kaufmann, San Francisco (1993)
2. Hornik, K., Stinchcombe, M., White, H., Auer, P.: Degree of approximation results for feed-forward networks approximating unknown mappings and their derivatives, NeuroColt Technical Report Series, NC-TR-95-004, 1-15 (1995)
3. Angiulli, F., Pizzuti, C.: Fast outlier detection in high dimensional spaces. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) PKDD 2002. LNCS (LNAI), vol. 2431, pp. 15–27. Springer, Heidelberg (2002)
4. Biesiada, J., Duch, W., Kachel, A., Mączka, K., Pałucha, S.: Feature ranking methods based on information entropy with Parzen windows. In: Proc. of Research in Electrotechnology and Applied Informatics, Katowice-Kraków, pp. 109–118 (2005)
5. Alexis, J., Jonsson, P., Jonsson, L.: Heating and electromagnetic stirring in a ladle furnace – a simulation model. ISIJ Int. 40(11), 1098–1104 (2000)
6. Wieczorek, T.: Intelligent control of the electric-arc steelmaking process using artificial neural networks. Computer Methods in Material Science 6(1), 9–14 (2006)
7. Siemens, A.G.: Optimization of the electrode control system with neural networks, pp. 1–8. Siemens Press (2003)
8. Wieczorek, T., Pyka, M.: Neural modeling of the arc-electric steelmaking process. In: Proc. of 9th Int. Conf. Research in Electrotechnology and Applied Informatics, Katowice, pp. 105–108 (2005)
9. Wieczorek, T., Blachnik, M., Mączka, K.: Building a model for time reduction of steel scrap meltdown in the electric arc furnace (EAF). General strategy with a comparison of feature selection methods. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 1149–1159. Springer, Heidelberg (2008)

10. Hall, M.A.: Correlation based feature selection for machine learning., PhD thesis, Dept. of Comp. Science, Univ. of Waikato, Hamilton, New Zealand (1998)
11. Kordos, M., Duch, W.: Variable step search algorithm for feedforward networks. Neurocomputing 71(13-15), 2470–2480 (2008)
12. Gallagher, M.: Multi-layer Perceptron Error Surfaces Visualization, Structure and Modeling., PhD Thesis, University of Queensland (2000)
13. Kordos, M., Duch, W.: A Survey of Factors Influencing MLP Error Surface. Control and Cybernetics 33(4), 611–631 (2004)
14. Schölkopf, B., Smola, A.J.: Learning with kernels. MIT Press, Cambridge (2002)
15. Eskander, G.S., et al.: Round Trip Time Prediction Using the Symbolic Function Network Approach, http://arxiv.org/ftp/arxiv/papers/0806/0806.3646.pdf
16. Setiono, R., Thong, J.: An approach to generate rules from neural networks for regression problems. European Journal of Operational Research 155(1), 239–250 (2004)

Trading Strategy in Foreign Exchange Market Using Reinforcement Learning Hierarchical Neuro-Fuzzy Systems

Marcelo F. Corrêa[1], Marley Vellasco[1], Karla Figueiredo[1], and Pedro Vellasco[2]

[1] Dept. of Electrical Engineering, PUC-Rio, Brazil
[2] Dept. of Structural Engineering, State University of Rio de Janeiro, RJ, Brazil

Abstract. This paper evaluates the performance of the new hybrid neuro-fuzzy model, Reinforcement Learning Hierarchical Neuro-Fuzzy System (RL-HNFP), in a trade decision application. The proposed model was tested with the Euro/Yen negotiated in Foreign Exchange Market. The main objective of the trading system is to optimize the resource allocation, in order to determine the best investment strategy. The performance of the RL-HNFP was compared with different benchmark models. The results showed that the system was able to detect long term strategies, obtaining more profitability with smaller number of trades.

1 Introduction

Financial markets are influenced by several factors, including economical, political and psychological issues. That is why its movements are extremely hard to forecast. During the last decades, research in this field has been growing very fast. These studies can be divided into two main areas: fundamental and technical analysis. The first is based on factors such as: company's financial statements; management and competitive advantages; competitors; and markets. On the other hand, technical analysis tries to predict the prices´ future direction based on the behavior of past market data [1].

Since the early 90s, neural networks models have been successfully applied to financial time series forecasting [2-7]. The most popular model is the Multi-Layer Perceptron (MLP) with backpropagation algorithm [8]. However, many researchers have suggested the use of reinforcement learning models to create trading strategies and to optimize asset allocation [9-12]. This type of learning works through an input-output mapping, built from the continuous environment interaction, trying to minimize a performance index [8].

One of the main advantages of RL in this kind of application is the fewer number of position changes or trades in the strategies. Generally, it results in less costs and higher profitability. For example, Neurier [9] states that a strategy to invest in the German Stock Index DAX based on RL attained better results than a MLP based model doing just 33% of the trades. The RL strategy kept the investment out of the market when there was not significant trend to follow.

Therefore, this paper evaluates the performance of the new Reinforcement Learning Hierarchical Neuro-Fuzzy System (RL-HNFP) [13] as an intelligent agent

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 461–468, 2009.

that learns the best trading strategy. The intelligent trading system was tested with the Euro/Yen exchange rate between 01/28/1999 and 05/18/2006. The performance of the RL-HNFP system was compared with different benchmark models: MLP neural network, Dynamic Regression, Box-Jenkins model, and the standard buy-and-hold strategy.

This paper is divided into four additional sections. Section 2 summarizes the RL-HNFP system, section 3 presents the proposed intelligent trading system and section 4 discusses the results obtained. Finally, the conclusions that have been achieved from this study are presented in last section.

2 RL-HNFP System

The RL-HNFP model is composed of various standard cells called RL-neuro-fuzzy BSP (RL-NFP). These cells are arranged in a tree hierarchical structure. The cells outputs in the lower levels are the consequents of higher levels cells.

An RL-NFP cell is a mini-neuro-fuzzy system that performs politree partitioning (2^n partitions, where n = number of input variables) of a given space, using the complementary membership functions described in equation (1) in each input dimension, where ρ(x) and μ(x) represent the low and high membership functions, respectively.

$$\mu(x) = \frac{1}{1+e^{-(a(x-b))}} \quad and \quad \rho(x) = 1 - \mu(x) \qquad (1)$$

where b is the sigmoid inflexion point and α is the sigmoid inclination at $x=b$.

The RL-NFP cell generates a precise output after defuzzification [14-15]. Each cell receives all inputs that are being considered in the problem. These inputs are inferred in the antecedents' fuzzy sets (ρ (x) and μ (x)). Figure 1(a) depicts a cell with two inputs – x_1 and x_2 -, where each partitioning is generated by the combination of the two membership functions, ρ (*low*) and μ (*high*) of each input variable, and is associated with a set of actions (a_1, a_2 ... a_t).

The consequents of the cell's partitions may be a singleton or the previous level stage output. Although the singleton consequent is simple, it is not previously known, since each consequent is associated with an action that has not been defined a priori.

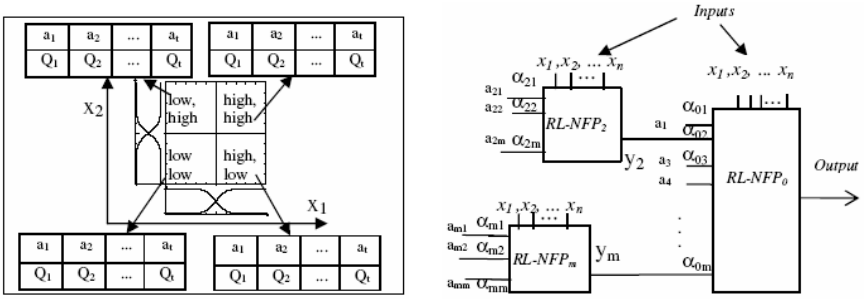


Fig. 1. (a) Internal representation of the RL-NFP cell with two inputs; (b) Example of architecture of the RL-HNFP model

Each partition has a set of possible actions, as shown in Figure 1(a), where each action is associated with a Q-value function. The Q-value is defined as being the sum of the expected values of the rewards obtained by the execution of action *a* in state *s*, in accordance with a policy π [16].

By means of the RL-based learning algorithm, one action of each poli-partition (for example, a_i, and a_q) is selected as the one that represents the desired behavior of the system whenever it is in a given state. Thus, the consequents are the actions that the agent must learn along the process.

The linguistic interpretation of the mapping implemented by the RL-NFP cell is given by fuzzy rules. Each rule corresponds to one partition generated by the politree partitioning and has the following structure:

If x_1 is low and …. x_n is high then y = $a_{i.}$

RL-HNFP models can be created based on the interconnection of basic cells described above. The cells form a hierarchical structure that results in the rules that compose the agent's reasoning. Figure 1(b) exemplifies this architecture.

Neuro-fuzzy learning process is generally divided into two parts: structure identification and parameter adjustment [17]. RL-HNFP performs these learning tasks in a single algorithm. Basically, each partition chooses an action from its set of actions; the resultant action is calculated by the defuzzification process and represents the action that will be executed by the agent's output. After the resultant action is carried out, the environment is read once again. This reading enables calculation of the environment reinforcement value that will be used to evaluate the action taken by the agent. The reinforcement is calculated for each partition of all active cells, by means of its participation in the resulting action. Thus, the environment reinforcement calculated by the evaluation function is backpropagated from the root-cell to the leaf-cells. Next, the Q-values associated to the actions that have contributed to the resulting action are updated, based on the SARSA algorithm [16]. More details can be found in [15].

3 The RL-HNFP Trading System

The RL-HNFP Trading System was built using the Euro/Yen exchange rate data, from 01/28/1999 to 05/18/2006. The series contained 2130 registers and the following variables were used: closing price in time t; difference between the 3 and 20-day moving averages; difference between the 5 and 20-day moving averages; 6-day momentum (difference between the price in t and t-6); 3, 5 and 20-day moving averages; open price; maximum and minimum prices; closing prices in t-1 and t-2; and the negotiated volume in t.

The least-square estimator (LSE) [18] was used to evaluate the relevance of each input. After that, just 8 variables remained at the model and were used to define the state in t. In order of relevance, the variables considered were: closing price in t, difference between the 3 and 20-day moving averages, difference between the 5 and 20-day moving averages, 6-day momentum, 5 and 3-day moving averages, closing prices in t-1 and maximum price in t.

The RL-HNFP objective is to learn the best action to be taken at each state t, after receiving, as environment reward, the signal of the price variation. To avoid sending undetermined signals and confusing the learning process, variations smaller than |0,3%| were changed to zero during the learning phase.

Trading strategies were composed by a sequence of three possible actions (*buy*, *sell* and *hold*). The *hold* signals were generated when the output belongs to a certain threshold interval. For example, if the threshold is defined between -0.02 and +0.02, and during 5 periods of time the RL-HNFP's output is 0.3, -0.2, 0.01, 0.25 and -0.01, then the system actions would be "buy", "sell", "hold sell position", "buy" and "hold buy position".

4 Results

The test set was composed of 213 patterns, corresponding to the interval 09/13/2005 to 05/18/2006. The results obtained can be observed in Table 1. The only cost considered was the spread cost estimated in 0.04% (difference between the bid and ask prices). As expected, by increasing the threshold, a higher number of hold signals are generated. Consequently, fewer trades are performed.

Without considering trading costs, the best performance was achieved by the strategy composed just by two actions (buy and sell) and no threshold. After 21 trades, the system reached profits of 7.52%, against 4.47% of a *buy-and-hold* benchmark strategy. However, if trading costs are included, the most profitable result was obtained by model 3. In this strategy, when the output of the RL-HNFP is between -0.02 and 0.02, the hold action is chosen and the previous position is maintained.

For a better understanding of the RL-HNFP Trading System's results, Figure 2 shows exactly when each decision of model 3 was taken. The value of +1 indicates a *buy position* and -1 a *sell position*. During a *buy position*, the investment return rate is the same as the Euro/Yen variation, and it can be positive or negative. In a *sell position*, no return is obtained. As can be observed, the system detected long term trends, without considering so much short term variations. This characteristic made the model more profitable during high long term periods, such as 09/13/2005 to 10/06/2005, 10/13/2005 to 11/18/2005, 11/27/2005 to 12/13/2005, and 01/08/2006 to 02/03/2006. The system indicated that the investor should be in a buy position during these intervals. Also, a graph with the evolution of a hypothetical initial investment of $100 is shown in Figure 3. The graph shows a consistent increase of the value invested during the time.

Table 1. RL-HNFP results: strategies with different actions and thresholds

Model	Actions	Threshold	# Trades	% Profit (without costs)	% Profit (with costs)
Buy and Hold	-	-	-	4,47%	4,47%
1	-1, 1	-	21	7.52%	6.64%
2	-1, 0, 1	±0.01	19	7.43%	6.67%
3	-1, 0, 1	±0.02	17	7.53%	6.86%
4	-1, 0, 1	±0.05	15	4.66%	4.08%
5	-1, 0, 1	±0.10	9	0.68%	0.35%
6	-1, 0, 1	±0.15	5	3.30%	3.14%

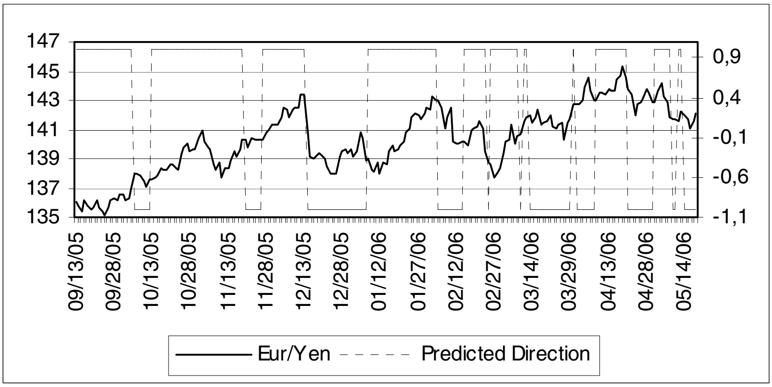


Fig. 2. RL-HNFP System strategy. +1 means a *buy position* and -1 a *sell position.*

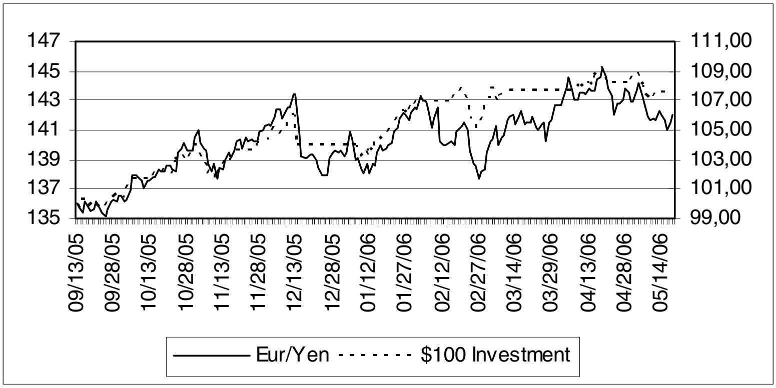


Fig. 3. Hypothetical $100 investment evolution using the RL-HNFP System

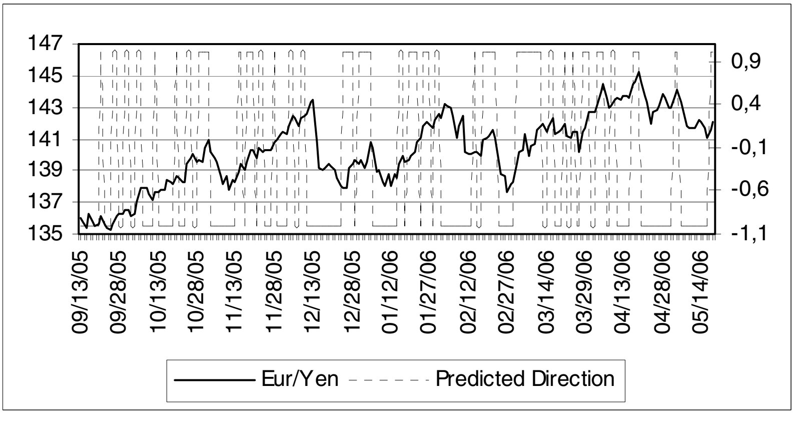


Fig. 4. MLP network strategy. +1 means a *buy position* and -1 a *sell position*.

Table 2. MLP results: strategies with different actions and thresholds

Model	Actions	Threshold	# Trades	% Profit (without costs)	% Profit (with costs)
Buy and Hold	-	-	-	4.47%	4.47%
MLP 1	-1, 1	-	104	7.56%	4.99%
MLP 2	-1, 0, 1	±0.01	98	7.83%	5.42%
MLP 3	-1, 0, 1	±0.02	84	7.80%	5.64%
MLP 4	-1, 0, 1	±0.05	70	4.28%	2.40%
BJ 1	-1, 1	-	102	-26.66%	-30.15%
BJ 2	-1, 0, 1	±0.01	56	-24.02%	-25.98%
BJ 3	-1, 0, 1	±0.02	18	-9.75%	-10.41%
BJ 4	-1, 0, 1	±0.05	2	-0.54%	-0.58%
DR 1	-1, 1	-	71	2.43%	-0.38%
DR 2	-1, 0, 1	±0.01	51	0.39%	-1.62%
DR 3	-1, 0, 1	±0.02	38	1.96%	0.45%
DR 4	-1, 0, 1	±0.05	6	0.86%	0.62%

For comparison purposes, a MLP network was developed. The test set and the input variables were the same for both systems. The MLP architecture was composed of a single hidden layer. The transfer function in all layers was the hyperbolic tangent. Three different topologies were created, with different number of neurons in the hidden layer (7, 9 and 11). The final architecture was selected by a validation set. The learning process was interrupted after 3000 epochs or by the early stopping strategy. Table 2 shows its results. The best performance was obtained by the model with 9 neurons in the hidden layer. So, this network was used for comparisons with the RL-HNFP Trading System.

Strategies based on Box-Jenkins ARIMA (BJ) and Dynamic Regression (DR) predictions were also developed using the same data. Results obtained are presented in Table 2.

The strategies created by BJ and DR were worst than the *buy-and-hold* benchmark. This was already expected, since both are linear methods. Additionally, since BJ is an autoregressive model, other inputs were not considered, which could to improve the forecast.

Analyzing the RL and MLP models, it can be verified that they provide a very distinct behavior. The number of trades of the RL-HNFP Trading System was significantly smaller. This can be explained by the different learning methods used in the MLP and RL-HNFP models. While reinforcement learning works through an input-output mapping, the supervised learning approximates the implicit function between the input variables and the output. The MLP model was also able to anticipate the price movement several times (see Figure 4), but the excessive number of operations made the system less profitable in long term trends. For example, during the intervals of 09/13/2005-11/02/2005 and 11/10/2005-12/13/2005 the gains were lower than in a buy-and-hold strategy. On the other hand, the strategy avoided loss in low trends, as it can be observed during the periods 11/02/2005-11/10/2005, 12/13/2005-12/26/2005, 01/05/2006-01/12/2006, 02/03/2006-02/13/2006 and 02/21/2006-02/27/2006.

However, during undefined trends, for example between 12/18/2005 and 01/18/2006, the MLP could take advantage of small variations. The MLP could also detect faster strong price falls, such as during the period of 12/13/2005-02/05/2006. Figure 4 shows actions taken by the system step by step.

5 Conclusions

In this paper, a Reinforcement Learning Hierarchical Neuro-Fuzzy System [14] was used for an investment decision problem. The main target was to optimize the resource allocation in defining the best Euro/Yen parity trade strategy. A MLP neural network was also developed, to be used as benchmark.

The results showed that the RL-HNFP Trading System could reach a better performance, doing just 20% of the trades done by the MLP model. When considering trading costs, it represents a significant profitability difference.

As future works, tests with other time series can be done to evaluate the RL-HNFP model in investment allocation problems. Also, it could be considered more than one asset to invest, using more actions to be optimized.

References

1. Damodaran, A.: Investment Valuation, 1st edn. John Wiley and Sons, Chichester (1996)
2. Amilon, H.: A Neural Network Versus Black-Scholes: A Comparison of Pricing and Hedging Performances. Journal of Forecasting 22, 317–335 (2003)
3. Azoff, E.M.: Neural Network Time Series Forecasting of Financial Markets. John Wiley & Sons Ltd., Chichester (1994)
4. Cheh, J., Weinberg, R.: An Application of an Artificial Neural Network Investment System to Predict Takeover Targets. Applied Business Research 15, 33–45 (1999)
5. Kutsurelis, J.E.: Forecasting Financial Markets Using Neural Networks: An Analysis of Methods and Accuracy. Master Thesis, Naval Postgraduate School (September 1998)
6. Refenes, A., et al.: Stock Performance Modeling Using Neural Networks: A Comparative Study with Regression Models 7(2), 375–388 (1994)
7. Yao, J., Tan, C.L.: Option Price Forecasting Using Neural Networks. Omega 28, 455–466 (2000)
8. Haykin, S.: Neural Networks — A Comprehensive Foundation. Prentice-Hall, Englewood Cliffs (1999)
9. Neunier, R.: Optimal asset allocation using adaptive dynamic programming. In: Advances in Neural Information Processing Systems, vol. 8. MIT Press, Cambridge (1996)
10. Moody, J., et al.: Performance Functions and Reinforcement Learning for Trading Systems and Portfolios. Journal of Forecasting 17, 441–470 (1998)
11. Lee, J.W.: Stock Price Prediction Using Reinforcement Learning. In: Proc. IEEE Int. Symposium on Industrial Electronics, vol. 1, pp. 690–695 (2001)
12. Lee, O.J., et al.: Adaptive stock trading with dynamic asset allocation using reinforcement learning. Information Sciences 176(15), 2121–2147 (2006)
13. Vellasco, M.M., Pacheco, M., Figueiredo, K., Souza, F.J.: Hierarchical Neuro-Fuzzy Systems - Part II, Encyclopedia of Artificial Intelligence. In: Rabuñal, J.R., Dorado, J., Pazos, A. (eds.) Information Science Referente (2008)

14. Figueiredo, K., Vellasco, M., Pacheco, M.: Hierarchical Neuro-Fuzzy Models based on Reinforcement Learning for Intelligent Agents. In: Cabestany, J., Prieto, A.G., Sandoval, F. (eds.) IWANN 2005. LNCS, vol. 3512, pp. 424–432. Springer, Heidelberg (2005)
15. Figueiredo, K., et al.: Reinf. Learning Hierarchical Neuro-Fuzzy Politree Model for Control of Autonomous Agents. In: 4th Int. Conf. on Hybrid Intelligent Systems (2004)
16. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
17. Jang, J.S.R.: ANFIS: Adaptive-network-based fuzzy inference system. IEEE Trans. on SMC 23(3), 665–685 (1993)
18. Chung, F.-L., Duan, J.-C.: On Multistage Fuzzy Neural Network Modeling. IEEE Trans. on Fuzzy Systems 8(2), 125–142 (2000)

Improving Multi Step-Ahead Model Prediction through Backward Elimination Method in Multiple Neural Networks Combination

Zainal Ahmad and Rabiatul Adawiah Mat Noor

School of Chemical Engineering, University Sains Malaysia, Engineering Campus,
Seri Ampangan,
14300 Nibong Tebal, Penang, Malaysia
chzahmad@eng.usm.my, rabiatul.adawiah84@gmail.com

Abstract. Combining multiple neural networks appears to be a very promising approach in improving neural network generalization since it is very difficult, if not impossible, to develop a solution that is close to global optimum using single neural network. In this paper, individual networks are developed from bootstrap re-sample of the original training and testing data sets. Instead of combining all the developed networks, this paper proposed backward elimination. In backward elimination, all the individual networks are initially aggregated and some of the individual networks are then gradually eliminated until the aggregated network error on the original training and testing data sets cannot be further reduced. The proposed techniques are applied to nonlinear process modeling and application results demonstrate that the proposed techniques can significantly improve model performance better than aggregating all the individual networks.

1 Introduction

Artificial neural networks have been increasingly used in developing nonlinear models in industry and model robustness is one of the main criteria that need to be considered when judging the performance of neural network models [1, 2]. Model robustness is primarily related to the learning or training methods and the amount and representativeness of training data [3]. Even though neural networks have significant capability in representing nonlinear functions, inconsistency of accuracy still seems to be a problem where neural network models may not perform well when applied to unseen data. Lack of robustness in neural network models is basically due to the overfitting and poor generalization of the models [4]. Among those approaches for improving neural network generalisation, the combination of multiple neural networks seems to be very effective. The individual networks in the multiple neural networks model the same relationship and are developed from different data sets and/or different training algorithms. They can also have different structures. Instead of choosing the single "best" neural network model, all the individual neural networks are combined. The main objective of this approach is to improve the generalization capability of the neural network models in such a way that it will guard against the failure of

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 469–476, 2009.

individual component networks. This is because of the fact that some of the neural networks will fail to deliver the correct results or output predictions due to network training converged to undesired local minima, over-fitting of noise in the data, or the limited training data set [5, 6].

In most of the reported works on aggregating multiple neural networks, all the developed individual networks are combined. However, some neural networks may not contribute to improving model prediction performance when combined with other networks. This could be due to several reasons, such as these networks severely overfit the data or the information captured by these networks has already been represented by other networks included in the aggregated network. Perrone and Cooper [7] suggests a heuristics selection method whereby the trained networks are ordered in terms of increasing mean squared errors (MSE) and only those with lower MSE are included in combination. However, combining these networks with lower MSE may not significantly improve model generalisation since these networks can be severely correlated. In this paper, backward elimination (BE) methods in statistical regression [8] are proposed for selective combination of neural networks.

Several literatures that employed the same method of combination can be found lately. Ahmad and Zhang [9] made a comparison between backward elimination and forward selection (FS) methods by using Bayesian method as combination method for the network ensemble. The ensemble networks have been tested on two case studies; 1) modelling of pH neutralization process, and 2) modelling of reactant concentration in an irreversible exothermic reaction process. These combination schemes showed their superiority compare to other schemes that combine all the networks. Partalas et al [10] also applied BE and FS methods to two types of classifiers; neural networks and support vector machine (SVM). The schemes are applied to a model of water quality prediction. This paper on the other hand is merely focus on BE method testing on two distinctive neural network structures; fixed and various structure and also a comparison with other combination schemes. The case study is pH neutralization process. The paper is organized as follows. Section 2 presents BE selective combination methods for aggregating multiple neural networks. Section 3 presents the case study to test the proposed techniques. Some results and discussions on the case study are given in Section 4. Finally, the last section concludes this paper.

2 Selective Combination of Multiple Neural Networks

Suppose that neural network models are to be developed from the data set $\{X, Y\}$, where $X \in R^{N \times p}$ is the input data, $Y \in R^{N \times q}$ is the output data, N is the number of samples, p is the number of input variables, and q is the number of output variables. To develop an aggregated neural network model containing n individual networks, the original data set can be re-sampled using bootstrap re-sampling with replacement to form n replications of the original data set [15]. The n replications can be denoted as $\{X_{(1)}, Y_{(1)}\}$, $\{X_{(2)}, Y_{(2)}\}$, …, $\{X_{(n)}, Y_{(n)}\}$, where $X_{(i)} \in R^{N \times p}$, $Y_{(i)} \in R^{N \times q}$, $i=1, 2, \ldots, n$. A neural network model can be developed on each of these replications, which can be partitioned into a training data set and a testing data set if cross-validation is used in network training and network structure selection. If the predictions of these n networks on the original data set are denoted as $\hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}n$, then the sum of squared errors (SSE) of the ith network can be calculated as

$$\mathrm{SSE_i} = \mathrm{trace}[(\mathrm{Y} - \hat{\mathrm{Y}}_\mathrm{i})^\mathrm{T}(\mathrm{Y} - \hat{\mathrm{Y}}_\mathrm{i})] \tag{1}$$

For the sake of simplicity in illustration, the simple average method is used in combining the selected networks. If all *n* networks are combined, then the aggregated network output is:

$$\hat{\mathrm{Y}} = \frac{1}{\mathrm{n}}\sum_{\mathrm{i=1}}^{\mathrm{n}} \hat{\mathrm{Y}}_\mathrm{i} \tag{2}$$

2.1 Backward Elimination

The BE approach was began with removing one network at a time from the aggregated network until sum square error (SSE) of the training and testing data cannot be further reduced. The network deleted at each step is such selected that its deletion results in the largest reduction in the aggregated network SSE on the training and testing data. The BE method is summarized as follows:

Step 1. Generate *n* replications of the original data set using bootstrap re-sampling, $\{X_{(1)}, Y_{(1)}\}$, $\{X_{(2)}, Y_{(2)}\}$, …, $\{X_{(n)}, Y_{(n)}\}$, and develop a neural network on each replication. Denote the prediction of the *i*th network on the original data set as $\hat{Yi}$. Calculate the SSE of these networks on the original data using Eq (1).

Step 2. Set j=1 and denote I as a set containing the indices of the networks currently included in the aggregated network and I= [1, 2, …, n]. Denote J as a set containing the indices of the networks currently deleted from the aggregated network and J=[], i.e. J is initially empty. Denote $\hat{Y}_{a,j}$ and SSE(j) as, respectively, the predictions and SSE of the aggregated network at stage j.

$$\mathrm{SSE(j)} = \mathrm{trace}[(\frac{1}{\mathrm{n}}\sum_{\mathrm{i}\in \mathrm{I}} \hat{\mathrm{Y}}_\mathrm{i} - \mathrm{Y})^\mathrm{T}(\frac{1}{\mathrm{n}}\sum_{\mathrm{i}\in \mathrm{I}} \hat{\mathrm{Y}}_\mathrm{i} - \mathrm{Y})] \tag{3}$$

Step 3. If n-j=0, then go to Step 5, else, execute the computation of the prediction by using the following chronicles of equations:

$j = j + 1$ and for $i \in I$, calculate prediction of the output by using Eq (4)

$$\hat{\mathrm{Y}}_{\mathrm{a,j}}^{(\mathrm{i})} = \frac{1}{\mathrm{n-j}}\sum_{\mathrm{l}\in \mathrm{I-i}} \hat{\mathrm{Y}}_\mathrm{l} \tag{4}$$

$$k = \arg\min_{i\in I} \mathrm{trace}[(\hat{Y}_{a,j}^{(i)} - Y)^T(\hat{Y}_{a,j}^{(i)} - Y)] \tag{5}$$

k is the argument for removing an individual network from the aggregated networks. After removing a network from the aggregated networks, SSE value for the current aggregated networks is computed using Eq (6)

$$SSE(j) = \mathrm{trace}[(\hat{Y}_{a,j}^{(k)} - Y)^T(\hat{Y}_{a,j}^{(k)} - Y)] \tag{6}$$

Step 4. If SSE(j) ≥ SSE(j-1), then go to Step 5, else remove the network, k from the aggregated networks, *I*. The process can be described as,

$I = I - k$ (i.e. remove *k* from *I*)

$J = [J, k]$ (indicate indices of deleted networks)

Then, the process is reiterated from Step 3 onwards.

Step 5. Stop

Selection of networks using BE method is executed by the mean of observing SSE on training and testing data sets. The network is removed until the SSE value cannot be further reduced. Firstly, SSE value for 20 networks for fixed and various structures are summed respectively. Then, one individual network from each cluster (fixed and various structure) is withdrawn and the SSE value of these 19 networks is observed. The withdrawn network is placed back with the 19 networks and a different network is again removed from the network cluster and again SSE value is observed. The process is repeated until one network that gives the least value of SSE is permanently removed from the network cluster. These steps can be counted as one cycle of BE selective combination process. The process is done until all 20 networks is removed one by one and then placed back inside the network group. The process of withdrawing networks is reiterated until the SSE value of both groups cannot be further reduced.

3 Case Study

The case study that was chosen is pH neutralization process. The neutralization process takes place in a CSTR and there are two input streams to the CSTR. One is acetic acid of concentration C_1 at flow rate F_1 and the other is sodium hydroxide of concentration C_2 at flow rate F_2 [11]. The mathematical equations of the CSTR can be found in reference [11]. To generate training, testing and validation data, multi-level random perturbations were added to the flow rate of acetic acid while other inputs to the reactor were kept constant.

The pH measurements were corrupted with normally distributed random noise with zero mean and a standard deviation of 0.2. The dynamic model representing the neutralization process is of the form:

$$\hat{y}(t) = f[\hat{y}(t-1), \hat{y}(t-2), u(t-1), u(t-2)] \quad (7)$$

where $\hat{y}(t)$ is the pH prediction in the reactor at time *t* and *u(t)* is the acid flow rates at time *t*. For long range predictions or multi-step-ahead predictions, the current and past model predictions are used to predict the future values of the model outputs:

$$\hat{y}(t) = f[\hat{y}(t-1), \hat{y}(t-2), \ldots\ldots., \hat{y}(t-n), u(t-1), u(t-2), \ldots\ldots u(t-m)] \quad (8)$$

where the model prediction, $\hat{y}(t-1)$ to $\hat{y}(t-n)$, are used in place of the process outputs, $y(t-1)$ to $y(t-n)$, to predict $\hat{y}(t)$ as shown for pH prediction in Eq (7).

In this case study, 20 networks with fixed identical structure and 20 networks with various structures were developed and the individual networks were trained by the Levenberg-Marquardt optimisation algorithm with regularisation and “early stopping”. In this study, fixed structure can be signified as a network with a fixed number of hidden neurons while various structures refer to networks with variety numbers of hidden neurons. The individual networks are single hidden layer feed forward neural networks. Hidden neurons use the sigmoid activation function whereas output layer neurons use the linear activation function. To cope with different magnitudes in the input and output data, all the data were scale to zero mean and unit standard deviation.

Accurate long range predictions are much more difficult to achieve than accurate one-step-ahead predictions due to the accumulation of the errors in the recursive predictions. To obtain long range predictions from an aggregated network, two types of network with output feedback schemes can be used but only feedback before combination of individual network is implemented in this study. To test the performance of the proposed selective combination schemes, the following combination schemes are investigated:

Median	:	Median of the individual networks;
Average	:	Average of all networks;
BE	:	Average of selected networks using the BE method.

Median of the individual networks can be described as the median of the SSE value of 20 networks, average of all networks is the SSE value of all networks that are combined using averaging method and BE refers as the SSE value of the BE selected networks that commingled with averaging method.

4 Results and Discussion

It is well known that the dynamics of pH is highly nonlinear. In this case study 20 networks with fixed number of hidden neurons (5) and 20 networks with varying number of hidden neurons (between 1 and 10) were developed. Again in the fixed structure, the number of hidden neurons was determined through cross validation. Fig. 1 shows the long range prediction performance of individual neural networks. It can be seen from Fig. 1 that the individual networks give inconsistent long range prediction performance on the training and testing data and on the unseen validation data. For example in Fig. 1 shows that network number 14 among the networks with various structures gives the worst performance on the training and testing data. However, its performance on the unseen validation data is quite good. This demonstrates the non-robust nature of individual networks.

Fig. 2 shows the SSE of long range predictions from aggregated neural networks with various structures. The aggregated networks under selective combination scheme give quite consistent prediction performance on the training and testing data and on the unseen validation data. This pattern was also observed for the fixed structure.

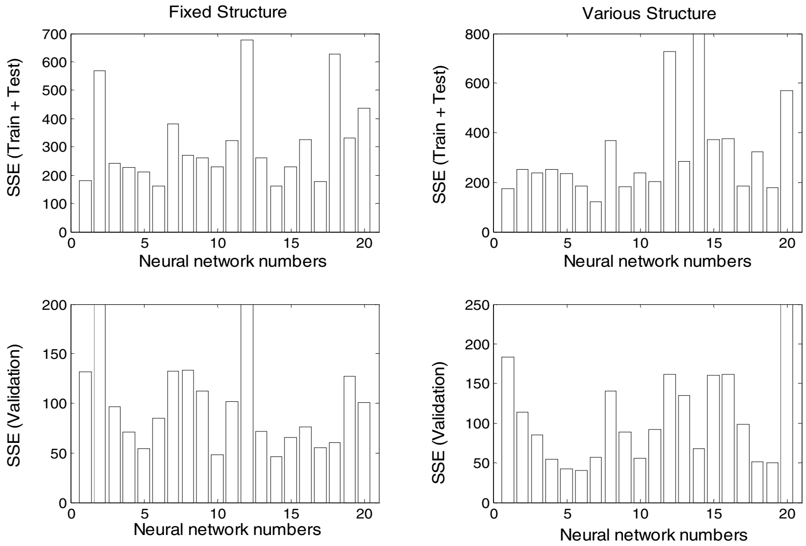


Fig. 1. SSE of long range predictions from individual neural networks in pH neutralization process

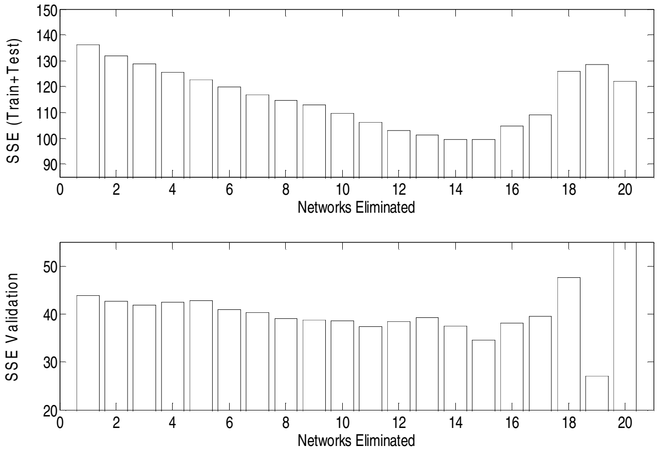


Fig. 2. SSE from aggregated neural networks with various structures in pH neutralization process

Table 1 gives the SSE on the unseen validation data of different combination schemes. It can be seen that the worse one of BE selective combination schemes gives better performance than combining all the networks and the median of individual networks. In the BE selection methods 5 networks (networks 1, 6, 11, 14, and 17) and 7 networks (networks 1, 5, 7, 11, 17, 18, and 20) were combined for fixed and various structures. The median of the individual network SSE on the unseen validation data for fixed and various structures are 90.44 and 90.52 respectively.

Table 1. Overall Results for pH Neutralization Process

Combination schemes		SSE on validation data
Median	Fixed structure	90.44
	Various structures	90.52
Average	Fixed structure	57.31
	Various structures	43.84
BE	Fixed structure	41.77
	Various structures	37.44

The best combination scheme in this case is “backward elimination method with fixed structures with feedback before combination” with an SSE of 37.44 on the unseen validation data.

5 Conclusions

Backward elimination methods for the selective combination of multiple neural networks are proposed in this paper in order to improve the model generalization performance. In the BE method, initially all individual networks are included in the aggregated network. Individual networks are then eliminated one at a time from the aggregated network until the aggregated network error on the original training and testing data cannot be further reduced. BE selective combination methods have shown their superiority compared to the combination of all networks and the median in this case study.

Acknowledgements. The author would like to thank USM (Grant No: 6035182 and 1001/JNC/AUPB001) for its continuous support and Dr Jie Zhang from Newcastle University, UK for providing the additional information on the Matlab$^{\text{TM}}$ simulation.

References

1. Wolpert, D.H.: Stacked Generalization. Neural Networks 5, 241–250 (1995)
2. Heartz, J.A., Krogh, A., Palmer, R.G.: Introduction to the Theory of Neural Network Computation. Addison-Wesley, Redwood City (1991)
3. Bishop, C.: Neural Networks for Pattern Recognition. Clarendon Press, Oxford (1995)
4. Caruana, R., Lawrence, S., Giles, L.C.: Overfitting in Neural Networks: Backpropagation, Conjugate Gradient and Early Stopping. Neural Information Processing System 13, 402–408 (2000)
5. Hashem, S.: Optimal Linear Combination. Neural Networks 10(4), 599–614 (1994)
6. Mc Loone, S., Irwin, G.: Improving Neural Networks Training Solution Using Regularization. Neurocomnputing 37, 71–90 (2001)
7. Perrone, M.P., Cooper, L.N.: When Networks Disagree: Ensembles Methods for Hybrid Neural Networks. In: Mammone, R.J. (ed.) Artificial Neural Networks for Speech and Vision, pp. 126–142. Chapman and Hall, London (1993)

8. Hagiwara, K., Kuno, K.: Regularization Learning and Early Stopping in Linear Networks. In: International Joint Conference on Neural Networks, pp. 511–516 (2000)
9. Ahmad, Z., Zhang, J.: Bayesian Selective Combination of Multiple Neural Networks for Improving Long-Range Predictions in Nonlinear Process Modelling. Neural Computing and Applications 14, 78–87 (2005)
10. Partalas, I., Tsoumakas, G., Hatzikos, E.V., Vlahayas, I.: Greedy Regression Ensemble Selection: Theory and an Application to Water Quality Prediction. Information Science 178, 3867–3879 (2008)
11. Mc Avoy, T.J., Hsu, E., Lowenthal, S.: Dynamics of pH in Controlled Stirred Tank Reactor. Ind. Chem. Process Des. Dev. 11, 68–70 (1972)

A Novel Adaptive Resource-Aware PNN Algorithm Based on Michigan-Nested Pittsburgh PSO

Kuncup Iswandy and Andreas König

Institute of Integrated Sensor Systems, University of Kaiserslautern,
67663 Kaiserslautern, Germany
{kuncup,koenig}@eit.uni-kl.de

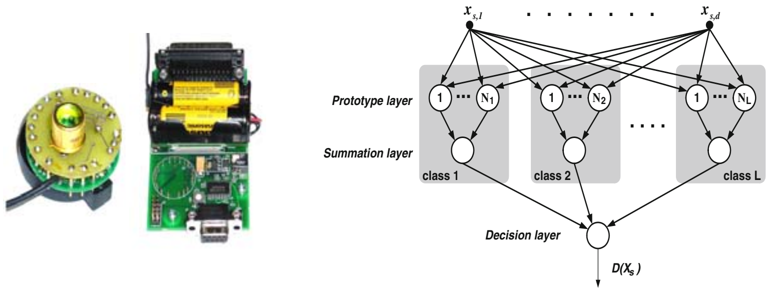
Abstract. The computational and power resource limitations applicable to intelligent sensor systems in mobile implementations have gained much attention for industrial and medical applications. Probabilistic Neural Networks (PNN) are one of a successful classifier used to solve many classification problems. Currently, in PNN all patterns of training set are used to estimate the probability density function (pdf) of a given class as the sum of isotropic Gaussian kernels. However, the computational effort and the storage requirement of PNN method will prohibitively increase as the number of patterns used in the training set increases. In this paper, we propose as a remedy an Adaptive Resource-Aware Probabilistic Neural Networks (ARAPNN) based on two optimization goals tackle by Particle Swarm Optimization (PSO), which are finding the proper position and number of prototypes (Michigan approach) as well as the best smoothing factor σ (Pittsburgh approach). Our proposed algorithm was be tested with five benchmark data sets. The results show that the ARAPNN is able to find solutions with significantly reduced number of prototypes that classify data with competitive or better accuracy than the original PNN and Nearest Neighbor classifiers.

1 Introduction

Nowadays, intelligent sensor systems find rapidly increasing industrial interests and applications. Wireless sensor network and Ambient Intelligence applications, such as Assisted Living tasks, are common examples of application fields. The designing issue of intelligent sensor systems demands to become more compact in size for easy distribution and mobile implementations. However, size and cost constraints will result in corresponding constraints on resources such as energy, memory, computational speed and bandwidth. As an example, an established sensor node configured as a wireless color sensor for color classification tasks is shown in Fig. 1(a), which stringently requires low complexity of pattern classifier due to limited hardware resource.

In this paper, we focus on one task of our research activity on systematic development of a methodology and framework for automated, application-specific design of intelligent integrated/embedded sensor systems. This work is under special constraint imposed by resource limitation in mobile and medical implementation. For decision making, e.g., in ambient intelligent systems, classification techniques are very important [8]. The goal of this paper is to contribute to the systematic design of lean but well performing classifiers for resource- and power-aware implementation.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 477–484, 2009.



(a) remote color sensor (left) and base-station (right)

(b) PNN model

Fig. 1. An example of a wireless sensor node (MICA) and general structure of PNN

There are numerous related works previously focused on lean classifiers. In non-parametric techniques, Hart proposed the pruning methods reduced the amount of data which has to be stored for the nearest neighbor classifier called Condensed Nearest Neighbor (CNN) [4]. Gates proposed a postprocessing step for the CNN pruning algorithm called Reduced Nearest Neighbor (RNN) [3]. The Restricted Coulomb Energy (RCE) is a three layer feedforward network, which gradually selects the prototypes in a only growing approach and adjusts their radii until satisfactory training [9]. The limitations of CNN, RNN, and RCE methods are: (1) their result strongly depends on presentation order of training set and (2) prototypes are selected from training without any adjustment. The work of Platt (1991) introduced Resource-Allocating Networks (RAN), which are related to RCE and Radial Basis Function networks (RBF). The RAN method allows to insert new prototypes for of Gaussian kernels, which will be adjusted as well as their centers by gradient descent technique in training [7]. This attractive method is hampered due to well-known gradient descent limitations. Improvements can be found, e.g., in Cervantes et al. proposing a new algorithm for nearest neighbor classification called Adaptive Michigan PSO, which can find less number of prototypes and adjust their positions [2]. These adjusted prototypes using AMPSO can classify the input patterns better than CNN, RNN, and RCE. The Michigan approach employed in this optimization technique is more efficient and requires much less computational effort than the standard PSO.

Currently, Probabilistic Neural Networks (PNN) still incorporate all patterns of the training set as classifier prototypes to estimate the probability density functions (pdf), which is not suitable for mobile implementation of intelligent sensor systems. In this paper, we propose an Adaptive Resource-Aware PNN (ARAPNN) algorithm based on Michigan-Nested Pittsburgh Particle Swarm Optimization, which can determine the proper number of prototypes required and adjusts them in their best positions. The Michigan approach is applied in similar way to AMPSO, which represent the individual particle as one of prototypes. This particle representation has advantages compared with original PSO, where particles have much lower dimension and less computational effort, also flexibility in growing and reducing the number of prototypes. The Pittsburgh approach represents the particles in similar way to original PSO. The smooth parameter of PNN is found by this nested optimization for each of iteration in the main algorithm.

Table 1. Four cases of different type of smoothing parameter

Type	for all classes	for all d features	number of σ
PNN-CC	common	common	1
PNN-IC	individual	common	L
PNN-CI	common	individual	d
PNN-II	individual	individual	L × d

2 Probabilistic Neural Networks (PNN) Model

The original PNN is a four-layer feedforward neural network, which is illustrated in Fig. 1(b). The first layer of PNN, which is designated as an input layer, accepts the input vectors to be classified. The nodes in the second layer, which is designated as a prototype unit, are divided into L groups with regard to their class affiliation. The active function used in second-layer node, the i^{th} kernel in the j^{th} group, is defined as a Gaussian basis function:

$$p_{ij}(x_k) = \frac{1}{(2\pi)^{\frac{d}{2}}\sigma_j^2}\exp(-\frac{\|x_k - c_{ij}\|^2}{2\sigma_j^2}), \tag{1}$$

where c_{ij} is the i^{th} training vector from class j, $j = \{1,...,L\}$, assumed as a center of a kernel function and σ_j a smoothing factor of class j. Let the number of prototype units for class j be N_j. The summation layer (third layer) estimates the class conditional probability density functions $f_j(x_k)$ of each class l_j. A competitive layer (final layer) makes the decision according to the Bayesian decision rule. Both third and fourth layer are defined as following:

$$f_j(x_k) = \frac{1}{N_j}\sum_{i=1}^{N_j} p_{ij}(x_k) \tag{2}$$

$$D(x_k) = \arg\max_j\{P(j)f_j(x_k)\}, \tag{3}$$

where $P(j)$ is the *a priori* class probability of class j.

Several authors [6,13] proposed the improvement of original PNN by applying different smoothing parameter summarized in Tabel 1. Yang and Chen [14] proposed Heteroscedastic PNN, where each particle has its own single smoothing factor σ. This idea is adopted in our approach of ARAPNN.

3 Adaptive Probabilistic Neural Networks

3.1 Michigan-Nested Pittsburgh PSO Algorithm

In our algorithm, we apply two optimization processes based on two different approaches of PSO algorithm, i.e. Michigan and Pittsburgh approaches. The Michigan approach placed as the main of optimization algorithm is used for finding the best position of prototypes and adjusting the number of prototypes. The Pittsburgh approach embedded inside the main algorithm as nested optimization procedure is used for obtaining the best smoothing factor for each set of prototypes. In the main algorithm of PSO, each particle is interpreted as a single prototype with its class affiliation, which is shown as a row vector:

$$x_i = (x_{i1}, x_{i2}, ..., x_{id}, c_i), \tag{4}$$

where d denotes the number of attributes (variables or features), x_i indicates the i^{th} prototype, and c_i denotes the information of class. This class does not evolve, but remains fixed for each particles. The attributes are normalized in the range [0,1], which are fixed the minimum and maximum values for the particles in all the dimensions. In the second stage of the nested PSO algorithm, the particles represent a set of solutions, which are smoothing factors (σ) of PNN. The overall stages are described below:

```
INITIALIZE: Insert N particles of each class from the training set. Randomize veloc-
ities of particles in search space. Randomize the smoothing factors $\sigma$.
REPEAT
    Check for particle reproduction and reduction.
    Collect which particles are in the competing and non-competing sets.
    FOR EACH particle
        Compute its local fitness.
        Find the closest particle in the non-competing and competing set.
        Update the particles.
    Find the optimal smoothing factors $\sigma$ applying nested PSO.
    Assign classes and compute the classification success in the training set.
    IF current global fitness larger than previous one THEN
        Record the particles' current positions as current best swarm.
UNTIL maximum iteration reached or saturation or success classification rate is 100%
Clean the useless particles from the best swarm without decreasing classification rate.
```

To influence the movement of each particle in each iteration, our algorithm adopts a concept of using dynamic neighborhood for the movement of each particle [2], where each particle may be influenced by different particles in both competing and non-competing sets. The competing set is a collection of particles, which have the most dominant pdf value for at least one pattern of same class. The non-competing set is a collection of particles, which have the most dominant pdf value for at least one pattern of different class.

3.2 Movement Equations of Particles

The movement concept of particles is based on sharing memory of their best positions and the particles neighborhood. There are four parts, i.e., the momentum part, the cognitive part, the attraction neighbor and the repulsion neighbor. The first two parts are similar to the original concept of PSO [5,11]. The attraction neighbor is used for guiding the search to the regions of a different class. This attracted neighbor is chosen according to the closest distance from non-competing particles in the population. The aim of using repulsion neighbor is to retain diversity and push each other to find new position of their class in different region of the search space. The idea of the repulsion in PSO have been introduced by other authors and can be found in [1,2,10].

The movement concept of particles in Michigan approach is described below:

$$\mathbf{v}_i^{t+1} = \omega \cdot \mathbf{v}_i^t + c_1 \cdot \mathbf{r}_1 \cdot (\mathbf{P}_i^t - \mathbf{x}_i^t) + c_2 \cdot \mathbf{r}_2 \cdot S_i \cdot (\mathbf{att}_i^t - \mathbf{x}_i^t) + c_3 \cdot \mathbf{r}_3 \cdot S_i \cdot (\mathbf{x}_i^t - \mathbf{rep}_i^t) \quad (5)$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \quad (6)$$

where $\mathbf{v}_i^{t+1}$ denotes as the i^{th} particle velocity in iteration t; $\mathbf{x}_i^t$ is the particle position; $\mathbf{P}_i^t$ is the best position achieved so far by the i^{th} particle; ω is the inertia weight c_1, c_2, and c_3 are constant weight factors; $\mathbf{r}_1$, $\mathbf{r}_2$, and $\mathbf{r}_3$ is random factors in the range [0,1]; $\mathbf{att}_i^t$ denotes the attraction neighbor for the i^{th} particle; $\mathbf{att}_i^t$ denotes the repulsion neighbor for the i^{th} particle; and S_i is an adaptation factor which is the inverse of the best local fitness of particle i.

3.3 Reproduction and Reduction of Prototypes

For adjusting the number of particles (prototypes) and their classes to the problem's requirements, two rules (i.e., reproduction and reduction) are applied. These rules are described below:

- A prototype has a chance to give birth to one prototype for each of classes in that set, when it has the highest pdf values among other prototypes for patterns from different classes. The chance of reproduction depends on the probability of reproduction P_a and their pdf value ($p_{ij}(x_k) > 0.5$). The new prototypes are placed into the population and their velocities are generated randomly.
- Particles can face deletion, if they do not give highest pdf value for any pattern. This deletion process depends on only the probability of reduction P_d, which grows linearly from 0 to the maximum P_d value in the last iteration.

3.4 Local and Global Objective Function

As in Michigan approach, each prototype μ_j has a local fitness value F_q, which measures its performance independently of the other prototypes. Introducing the sets S_q^r and S_q^w, which include the patterns x_k^r and x_k^w for which μ_q provides the largest pdf value of all prototypes, and also introducing P_r and P_w which are the corresponding accumulated pdf values $p(x_k^r)$ and $p(x_k^w)$, respectively, this local fitness function is computed as:

$$F_q = \begin{cases} 0 & if S_q^r = S_q^w = \emptyset \\ \frac{P_q^r}{N_r} + 1 & if S_q^w = \emptyset \\ \frac{P_q^r - P_q^w}{N_r + N_w} + 1 & \text{otherwise,} \end{cases} \quad (7)$$

with

$$P_q^r = \sum_k p(x_k^r) \qquad and \qquad P_q^w = \sum_k p(x_k^w). \quad (8)$$

$$S_q^r = \bigcup_k \{x_k^r\} \qquad and \qquad S_q^w = \bigcup_k \{x_k^w\} \quad (9)$$

Table 2. Four benchmark data sets used in the experiments

Name	Patterns	Features	Classes	Class Distribution
Bupa	345	6	2	200 / 145
Diabetes	768	8	2	500 / 268
Wine	178	13	3	59 / 71 / 48
Thyroid	215	5	3	150 / 35 / 30
Glass	214	9	6	70 / 76 / 17 / 13 / 9 / 29

Here, the index k is not contiguous, but assumes index value of those patterns, that cause maximum pdf value generation by prototype μ_q on presentation. The global function deals with two functions, i.e., maximizing the classification accuracy and minimizing the number of prototypes. The global function is defined as:

$$F_{global}(t) = w_1 \cdot Q_p(t) + w_2 \cdot \frac{T_p - N_p(t)}{T_p}, \tag{10}$$

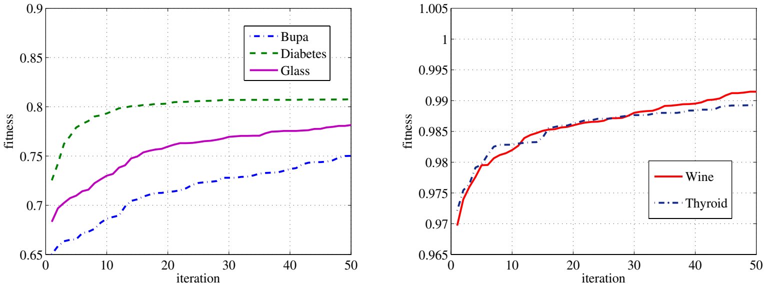
where w_1 and w_2 are weighting factors, Q_p is the classification accuracy, T_p is the total number of examples in training set, and N_p is the number of prototypes.

4 Experiments and Results

In this section, we perform experimentation on five well-known real problems collected from the UCI Machine Learning Repository. The five data sets are summarized in Table 2. All the data sets have real-valued features and are normalized in the range [0, 1]. We divided randomly each data set into 60% for training set and 40% for test set proportionally. In all the experiments, we repeated 20 runs for each of data sets and computed the average results.

In the main algorithm, the swarm parameters with regard to computation time were set as follows: for each class N = 3 patterns are randomly selected as initial particles and the number of iteration was set to 50. The ω was set to 0.1; c_1, c_2, and c_3 were set 0.35, 0.35, and 0.1, respectively. The probability of addition P_a was set 0.01 with linearly decreasing to 0 in the end of iteration. The probability of addition P_d was set 0.01 for final iteration with linearly increasing from 0 as initial value. For the nested optimization, the parameter settings was similar way to original PSO [11] with maximum iteration set to 30.

The results of ARAPNN on all the data sets are summarized in Table 3, where we compared our results with the results of original PNN with four different type of smoothing factor σ and other classifiers, i.e., 1-NN, 3-NN, RNN, and SVM. The success rate in terms of accuracy for ARAPNN is competitive and even better than the results of 1-NN, 3-NN, and RNN and four types of PNN for Bupa, Thyroid and Diabetes problems. Only in Wine dataset, PNN-IC and PNN-II achieved slightly higher results than ARAPNN. For all data set, SVM showed superior performance over all other classifiers including ARAPNN but substantially higher cost. The performance of ARAPNN on five benchmark data sets in training is shown in Fig. 2. In Table 4, we compare the complexity of three classifier (i.e., RNN, SVM and ARAPNN) measured



(a) (b)

Fig. 2. Averaged fitness of 20 runs: (a) Bupa, Diabetes, and Glass and (b) Wine and Thyroid

Table 3. Averaged (standard deviation) classification accuracy (%) with five different benchmark data sets using 20 repetitions

Data set	Bupa	Diabetes	Wine	Thyroid	Glass
1-NN	61.63 (4.10)	69.54 (2.45)	95.70 (1.41)	96.05 (2.68)	67.62 (3.85)
3-NN	62.32 (4.00)	72.74 (1.68)	95.70 (1.61)	93.72 (1.82)	65.23 (3.38)
RNN	59.06 (4.15)	65.64 (2.52)	93.24 (2.36)	94.65 (1.86)	64.71 (3.69)
PNN-CC	62.93 (4.82)	73.92 (2.00)	95.14 (2.12)	95.87 (1.66)	66.10 (3.50)
PNN-IC	62.78 (4.47)	74.09 (2.50)	96.62 (1.96)	95.47 (1.88)	66.80 (2.96)
PNN-CI	61.20 (3.54)	75.26 (2.25)	95.28 (2.01)	95.81 (1.66)	67.79 (3.89)
PNN-II	61.70 (2.70)	75.50 (1.72)	96.48 (2.22)	95.06 (2.79)	67.91 (5.31)
SVM	66.67 (3.90)	75.91 (1.85)	96.90 (1.69)	96.57 (1.62)	68.13 (2.98)
ARAPNN	**64.49 (3.16)**	**75.57 (1.69)**	**96.41 (1.68)**	**96.40 (1.96)**	**67.70 (2.15)**

Table 4. The averaged (standard deviation) number of prototypes

Data set	Bupa	Diabetes	Wine	Thyroid	Glass
RNN	123.75 (5.94)	233.20 (9.08)	19.05 (3.98)	20.55 (2.89)	65.25 (4.36)
SVM	140.20 (8.56)	237.65 (7.65)	35.75 (2.51)	27.30 (1.66)	136.85 (4.49)
ARAPNN	**41.05 (7.39)**	**166.45 (8.69)**	**18.85 (3.80)**	**12.40 (2.74)**	**28.24 (4.04)**

in terms of the average number of prototypes. The results in Table 4 show that ARAPNN has lowest complexity of all competitors.

5 Conclusion

The purpose of this paper is to develop an effective algorithm as a contribution for automated application-specific design of intelligent integrated/embedded sensor systems with focusing on mobile sensor implementations. The requirement of lean method but still well performing solutions for resource-aware system implementation is pursued. The presented study was related to lean, yet well performing classification algorithms employing and extending PNN. In our new ARAPNN, there are two matters, which have to be optimized, i.e., number of prototypes and the smoothing factor. We propose an efficient algorithm based on Particle Swarm Optimization applying Michigan-nested Pittsburgh approach. The Michigan approach takes each particle to represent a single prototype, which produce a search space of low dimension. It is opposite to original PSO, which straightforward encodes a set of prototypes in each particle that might hinder the swarm success due to high dimension. The optimization of smoothing factor is then applied by nested Pittsburgh approach, which is standard PSO. In addition, particle reproduction and deletion give a great contribution for growing and reducing of swarm size. The experiments show that our ARAPNN can produce a small representative set of prototypes and still perform well and achieve results close to SVM, but substantially lower cost, showing that ARAPNN offers excellent trade-off of resource requirements and performance. In future work, we will have more extensive statistical analysis (k-fold cross-validation / leave-one-out) and consider to adapt the approach to other kernel-based methods, e.g., RCE, RBF and SVM.

References

1. Blackwell, T.M., Bentley, P.J.: Don't Push Me! Collision-Avoiding Swarms. In: Proc. of the IEEE CEC, pp. 1691–1696 (2002)
2. Cervantes, A., Galván, I., Isasi, P.: An Adaptive Michigan Approach PSO for Nearest Prototype Classification. In: Mira, J., Álvarez, J.R. (eds.) IWINAC 2007, Part II. LNCS, vol. 4528, pp. 287–296. Springer, Heidelberg (2007)
3. Gates, W.: The Reduced Nearest Neighbor Rule. IEEE TIT 18, 431–433 (1972)
4. Hart, P.E.: The Condensed Nearest Neighbor Rule. IEEE TIT 14, 515–516 (1968)
5. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. IEEE ICNN 4, 1942–1948 (1995)
6. Montana, D.: A Weighted Probabilistic Neural Networks. ANIPS 4, 1110–1117 (1992)
7. Platt, J.: A Resource-Allocating Network for Function Interpolation. NC 3, 213–225 (1991)
8. Powner, E.T., Yalcinkaya, F.: Intelligent Sensors: Structure and System. Sensor Review 15(3), 31–35 (1995)
9. Reilly, D.L., Cooper, L.N., Elbaum, C.: A Neural Model for Category Learning. Bio. Cybernetics 45, 35–41 (1982)
10. Riget, J., Vesterstrøm, J.S.: A Diversity-Guided Particle Swarm Optimizer - the ARPSO. Technical report no. 2002-02, EVALife, Dept. of Computer Science, Univ. of Aarhus (2002)
11. Shi, Y., Eberhart, R.C.: A Modified Particle Swarm Optimizer. In: Proc. of the IEEE ICEC, pp. 69–73 (1998)
12. Specht, D.F.: Probabilistic Neural Networks. Neural Networks 3, 525–532 (1990)
13. Tang, W.H., Goulermas, J.Y., Wu, Q.H., Richardson, Z.J., Fitch, J.: A Probabilistic Classifier for Transformer Dissolved Gas Analysis with a Particle Swarm Optimizer. IEEE Trans. Power Delivery 23(2), 751–759 (2008)
14. Yang, Z.R., Chen, S.: Robust Maximum Likelihood Training of Heteroscedastic Probabilistic Neural Networks. NN 11(4), 739–748 (1998)

Imputation of Missing Data Using PCA, Neuro-Fuzzy and Genetic Algorithms

Nthabiseng Hlalele[1], Fulufhelo Nelwamondo[2], and Tshilidzi Marwala[1]

[1] School of Electrical and Information Engineering, University of the Witwatersrand
Private Bag 3, Wits, 2050
nthabiseng.hlalele@students.wits.ac.za, t.marwala@ee.wits.ac.za
[2] School of Arts Sciences, Harvard University, Hollyoke Center 350, Cambridge,
Massachusetts, 023138
nelwamon@fas.harvard.edu

Abstract. This paper presents a method of imputing missing data that combines principal component analysis and neuro-fuzzy (PCA-NF) modeling in conjunction with genetic algorithms (GA). The ability of the model to impute missing data is tested using the South African HIV sero-prevalence dataset. The results indicate an average increase in accuracy from 60 % when using the neuro-fuzzy model independently to 99 % when the proposed model is used.

1 Introduction

The missing data problem is a widely researched topic that has a huge impact on any field that requires the analysis of data in order to make a decision or reach a specific goal [1-7]. A number of methods have been investigated and implemented in order to deal with this problem, especially in large databases that require computational analysis [1], [2], [3]. Traditionally, ad hoc methods have been used when dealing with missing data; these include mean substitution and the deletion of all data entries that contain missing variables. Although easy to implement, these methods often lead to loss of data resulting in a more biased database. This has led to the development of more advanced regression techniques and likelihood based approaches such as expectation maximization (EM). Auto-associative neural networks (AANN) in conjunction with genetic algorithms have been employed and modified to improve the accuracy of computational methods in imputing missing data [1, 6]. This paper adds to this knowledge by employing principal component analysis, neuro-fuzzy modeling and genetic algorithms to impute missing data in the HIV sero-prevalence dataset. The backgrounds of the missing data problem, neuro-fuzzy computing and PCA are presented. The PCA-NF-GA method along with its testing data and measures are then presented followed by the results and discussions.

2 Background

The background of the missing data problem and its mechanisms is presented. Neuro-fuzzy networks, PCA and genetic algorithms are also briefly discussed.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 485–492, 2009.

2.1 Missing Data

Missing data estimation, like any other data analysis method, depends on the knowledge of how the data are missing. Three mechanisms of missing data have been documented [1-2], [4], [8-9] which include missing completely at random (MCAR), missing at random (MAR) and missing not at random (MNAR) also called the non-ignorable case. MCAR occurs when the probability of the missing data variable is independent of the variable value itself or on any other values in the database. MAR occurs when the probability of the missing data variable is dependent on other variables in the database but not on the value of the variable itself. This means that there exists some complex relationship between the observed and missing data; i.e. the observed data can be used to approximate the missing data. When data are MNAR, the probability of the missing data variable is related to the missing data, this means that missing data variables cannot be predicted from the observed database. Dealing with this type of missing data is difficult and may involve imputing the missing variables based on external data not within the database [9]. Because it is often difficult to determine what mechanism brought about the missing data, artificial intelligence methods have been investigated to solve the missing data problem irrespective of its missing mechanism [2, 7].

2.2 Neuro-Fuzzy Computing and Genetic Algorithm

The neuro-fuzzy architecture integrates the use of neural networks, which identify interrelationships and patterns in numerical datasets, and fuzzy systems, that incorporate expert knowledge and perform decision making [10]. This results in an inference system (as a result of fuzzy rules) that has the ability to learn and adapt through its environment (as a result of neural networks). A conventional fuzzy system uses expert knowledge to produce a linguistic rule base and reasoning mechanism for decision making. If artificial neural networks, together with an optimization technique, are incorporated into the fuzzy model to automatically tune the fuzzy parameters (antecedent membership functions and parametric consequent models), then the product is a neuro-fuzzy inference system [10-12]. In this paper, the Takagi-Sugeno fuzzy inference system is used because of its ability to generate fuzzy rules from an input-output dataset thus encapsulating expert knowledge that is otherwise lost when using traditional neural networks or autoencoders. The training of this system is performed on a structural and parametric level. Structural tuning is used to find the appropriate number of rules and partitioning of the input space by minimizing the sum of squares error function between the predicted and target values during training and the parametric tuning determines the optimum antecedent membership functions and consequent parameters. Genetic algorithms (GA) are inspired by the theory of evolution involving genetic processes such as mutation, selection and cross over [13]. The Genetic Algorithms Optimization Toolbox (GAOT) is used to optimize the value of an evaluation function [14]. When AANN, which recall the input, are used for missing data imputation, the GA attempts to minimize the error between the output and the input data.

2.3 Principal Component Analysis

Principal Component Analysis is a statistical technique used for data dimension reduction and pattern identification in high dimensional data [15]. The PCA orthogonilises the components of the input vectors to eliminate redundancy in the input data thereby exploring correlations between samples or records. It then orders the resulting components such that the components with the largest variation come first. The compressed data (mapped into i dimensions) is presented by:

$$Y_{j\times i} = X_{j\times k} \times PCvector_{k\times i}\,. \tag{1}$$

Where the principal component vector, $PCvector$ is presented by the eigenvectors of the i largest eigenvalues of the covariance matrix of the input $X_{j\times k}$ with k dimensions and j set of records $(i \le k)$.

3 Proposed Method

The method used to impute the missing data is presented. First the dataset as well as the data preprocessing are presented followed by the method used to impute the missing data.

3.1 HIV Sero-Prevalence Data

This dataset was obtained from the South African antenatal seroprevalence survey of 2001 [16]. The data for this survey were collected from questionnaires answered by pregnant women visiting selected public clinics in South Africa and only women undertaking in the study for the first time were allowed to participate. This dataset has been used to investigate the effect that demographic information has on the HIV risk of an individual [17]. The data attributes used in this study are the HIV status, Education level, Gravidity, Parity, the Age of the Mother (pregnant woman) and the Age of the Father responsible for the most recent pregnancy. The HIV status is represented in binary form, where 0 and 1 represent negative and positive respectively. The education level indicates the highest grade successfully completed and ranges between 0 and 13 with 13 representing tertiary education. Gravidity is the number of pregnancies, successful or not, experienced by a female, and is represented by an integer between 0 and 11. Parity is the number of times the individual has given birth and multiple births (e.g. twin births) are considered as one birth event. It is observed from the dataset that the attributes with the most missing values are the age of the father (3972 missing values), the age of the mother (151 missing values) and the education level (3677 missing values) of the pregnant woman. In situations where an attribute is missing in the questionnaire, it is almost impossible to retrieve this information from the woman who supplied it due to the anonymity of the study. It is for this reason that missing data imputation methods are employed.

3.2 Data Preprocessing

When fitting a model in order to solve a problem, it is necessary to prepare the data such that the essence of the data is captured by the proposed model. First the data entries are normalized within the range [0 1] in order to implement the neuro-fuzzy model and, secondly, the data entries that contain logical errors are removed. The data is then evaluated in order to see which attributes contribute the most outliers as shown in figure 1.

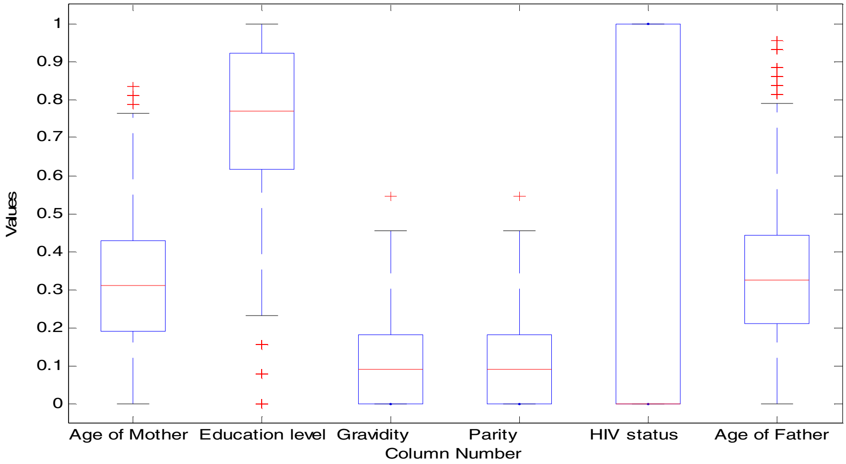


Fig. 1. HIV dataset outliers per attribute

It is important to remove outliers because they often represent misplaced data points which result in longer training times and models that perform poorly. The crosses in the figure represent the outliers that are present as a result of each attribute, it is clear that the attributes that are more likely to be missing in the dataset i.e. the age of the mother, the education level and the age of the father, also produce the most outliers. The data is then arbitrarily partitioned into 9745 datasets to train the model and 2462 testing datasets.

3.3 Proposed Method and Simulation

During training, PCA is employed to orthogonalize the data ensuring that the model is better trained. Performing PCA on the input data results in orthogonal data that has variance percentages as illustrated by figure 2. It is clear that a greater percentage (75 %) of the variance in the principal components can be attributed to the first two principal components illustrating the orthogonality of the training data.

The compressed data is then used during training to model all three of the data attributes that are likely to be missing. Figure 3 represents the proposed missing data imputation model.

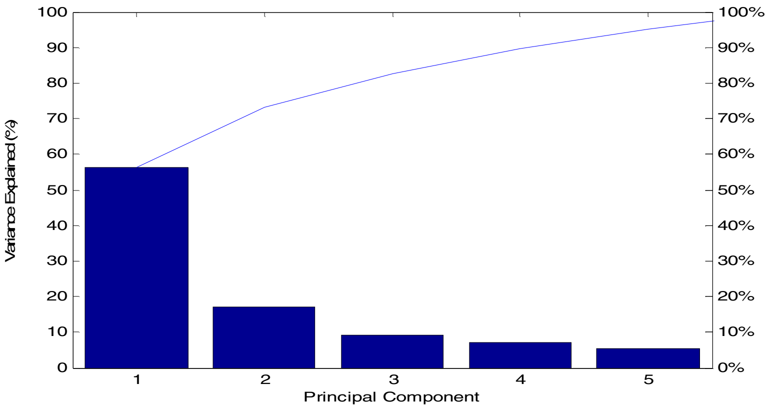


Fig. 2. Variance of the training input data principal components

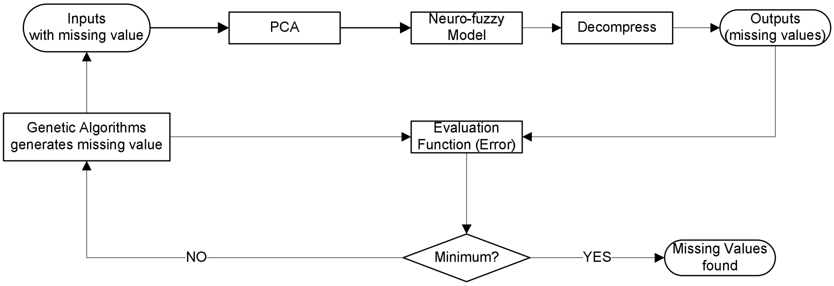

Inputs with missing value
PCA
Neuro-fuzzy Model
Decompress
Outputs (missing values)
Genetic Algorithms generates missing value
Evaluation Function (Error)
NO
Minimum?
YES
Missing Values found

Fig. 3. Flowchart of the proposed model for imputing missing data

When an input matrix that contains a missing variable is fed into the proposed model, it is compressed and a previously trained neuro-fuzzy model is used to impute the variable. The error between the imputed missing values and those generated by the GA is used as the evaluation function that needs to be minimized. If the error is not minimal, the GA generates a new missing value that will minimize the error.

4 Results and Discussions

4.1 Results

A test sample is first evaluated using the neuro-fuzzy model on its own to impute the father's age without compressing the data. The father's age is chosen as the test experiment because it is the field with the most missing values. The results of this test experiment are shown in Figure 4. These results indicate that the neuro-fuzzy model is unable to impute the missing data with great accuracy. The age of the father can only be imputed with an accuracy of 60 % within a 10 % margin. At first glance these results might seem satisfactory due to the fact that people are usually classified within a certain age group (e.g. anyone from the ages of 13 and 19 years is considered a

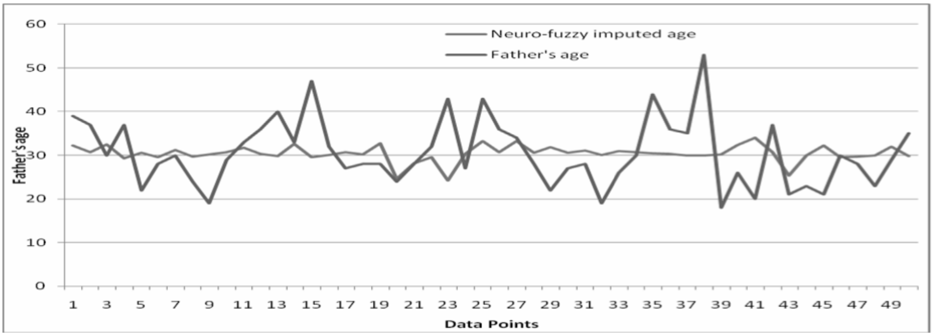


Fig. 4. Neuro-Fuzzy imputation of the father's age

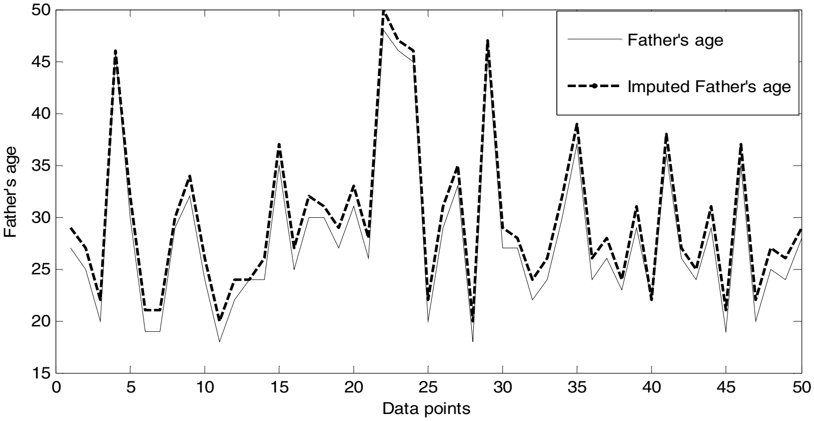


Fig. 5. Proposed model's imputation of the father's age

teenager etc), however, the measure used indicates poor performance since an older person is given a larger margin of error than a younger person (10 % of 50 is 5 whereas 10 % of 16 is 1.6). Similar results are observed when the N-F system and the GA are used independently without the compression of the data where an accuracy of 65 % is observed when the age of the father is estimated within a 10 % margin.

By employing a hybrid method such as the one proposed here, the accuracy of the imputation is expected to increase because of the ability of the hybrid system to capture hidden relationships in the dataset. Following the test experiment, the proposed model is then implemented to impute the missing data yielding the results shown in figure 5 for the age of the father. There is obvious correlation between the imputed and actual age of the father.

The imputation results of the age of the mother also correlate quite well as indicated by table 1. The accuracy of the mother or father's age is measured within 1 and 2 years. The imputed education level of the mother has no correlation at all with the actual level indicating a low accuracy value measured within 1, 2 and 5 grades. This, for example, means that the system has 98.99 % accuracy in estimating a woman's age within ± 1 year.

Table 1. Percentage of data that are correctly imputed

Attribute	Exact Accuracy (within 0)	Accuracy within 1	Accuracy within 2	Accuracy within 5
Mother's age	43.638 %	98.99 %	100 %	-
Father's age	6 %	37.67 %	99.9 %	-
Education level	2 %	9 %	13 %	26.7 %

4.2 Discussion and Conclusion

From Figure 1, it is deducible that the attributes with the most outliers are also the attributes that are likely to be missing. This is because both the missing data problem and the problem of outliers contain extreme values that provide erroneous information and modelling. This type of information is useful when building models that impute missing data. When the results in Figure 4 are compared with that of Figure 5 (imputation of the age of the father), it is clear that using the hybrid method provides better accuracy in imputing the age. When the ability of the system to impute the age of the mother is compared to Figure 1, it is deducible that the less varied the outliers of a variable (such as the case with the age of the mother), the higher the imputation accuracy of the model for that variable. The inverse can thus be the reason for the low accuracy of the system to impute the education level which suggests that other methods be looked into for imputing variables that have varied outlier models.

References

1. Ssali, G., Marwala, T.: Estimation of Missing Data using Computational Intelligence and Decision Trees (2007) arXiv 0709.1640
2. Nelwamondo, F., Mohamed, S., Marwala, T.: Missing Data: A comparison of Neural Networks and Expectation Maximisation Techniques. Current Science 93, 1514–1521 (2007)
3. Tresp, V., Neuneir, R., Ahmad, S.: Efficient Methods for Dealing with Missing Data in Supervised Learning. In: Tesauro, G., Touretzky, D.S., Leen, T.K. (eds.) Advances in Neural Information Processing Systems, vol. 7. MIT Press, Campbridge (1995)
4. Little, R., Rubin, D.: Statistical Analysis with Missing Data. John Wiley, New York (1987)
5. Schafer, J., Olsen, M.: Multiple imputation for Multivariate Missing Data Problems: A Data Analyst Perspective. Multivariate Behavioural Research 33, 545–571 (1997)
6. Abdella, M., Marwala, T.: The use of Genetic Algorithms and Neural Networks to Approximate Missing Data in Database. Computing and Informatics 24, 577–589 (2005)
7. Nelwamondo, F., Marwala, T.: Rough Sets Theory for the Treatment of Incomplete Data. In: Proceedings of the IEEE International Conference on Fuzzy Systems, pp. 338–343 (2007)

8. Rubin, D.: Multiple Imputations in Sample Surveys-A Phenomenological Bayesian Approach to Nonresponse. In: Proceedings of the Survey Research Methods Section of the American Statistical Association, pp. 20–34 (1978)
9. McKnight, P., McKnight, K., Sidani, S., Figuredo, A.: Missing data: A gentle Introduction. Guilford Press, New York (2007)
10. Jang, J.-S.R., Sun, C.-T., Mizutan, E.: Neuro-Fuzzy and Soft Computing; A Computational Approach to Learning and Machine Intelligence. Prentice-Hall, Upper Saddle River (1997)
11. Bontempi, G., Bersini, H.: Now Comes the Time to Defuzzify Fuzzy Models. Fuzzy Sets and Systems 90(2), 161–170 (1997)
12. Bontempi, G., Bersini, H., Birattari, M.: The Local Paradigm for Modeling and Control: From Neuro-Fuzzy to Lazy Learning. Fuzzy Sets and Systems 121(1), 59–72 (2001)
13. Goldberg, D.: Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley, Boston (1989)
14. Houck, C., Joines, J., Kay, M.: A Genetic Algorithm for Function Optimization: A Matlab Implementation. North Carolina State University, Raleigh, NC, Tech. Rep. NCSUIE-TR-95-09 (1995)
15. Jollife, I.T.: Principal Component Analysis. Springer, New York (1986)
16. South African Department of Health: HIV and Syphilis Sero-Prevalence Survey of Women Attending Public Antenatal Clinics in South Africa, http://www.info.gov.za/view/DownloadFileAction?id=70247
17. Betechuoh, B.L., Marwala, T., Manana, J.V.: Computational Intelligence for HIV Modeling. In: Proceedings of the International Conference on Intelligent Engineering Systems, pp. 127–132 (2008)

Feature Selection Method with Multi-Population Agent Genetic Algorithm

Yongming Li[*] and Xiaoping Zeng

College of Communication Engineering, Chongqing University, Chongqing, China 400030
lymcentor924924@gmail.com

Abstract. The multi-population agent genetic algorithm (MPAGAFS) for feature selection is proposed. The double chain-like agent structure is introduced to enhance the diversity of population. The structure can help to construct multi-population agent GA, thereby realizing parallel searching for an optimal feature subset. The experimental results show that the MPAGAFS can not only be used for serial feature selection but also parallel feature selection with satisfying precision.

1 Introduction

Because of its implicit parallelism, genetic algorithm (GA) is an effective search algorithm for finding near–optimal solutions in complex search spaces of feature selection. However, the traditional GA is too dependent on initial population; the simple genetic operators lead to falling into local optimal feature subset more easily.

In order to improve its performance, many researchers proposed some modified GAs. These improvements relate to genetic operators, population size, population structure, selection strategy, and so on [1-2]. Leung and Wang proposed an improved genetic algorithm- the orthogonal genetic algorithm with quantization [1]. M. Srinivas el proposed an adaptive genetic algorithm [2]. Il-Seok Oh proposed hybrid genetic algorithm putting genetic algorithm and local optimization method together for improvment [3]. Agent is one unit can interact with the environment around it and is driven by certain purposes. Weicai Zhong [6] proposed a lattice-like agent population structure to solve the problem. Currently, feature selection problems always are large scale, apparently one processor is not enough for running feature selection to meet the requirement of time cost. Therefore, parallelism is needed to be introduced into genetic algorithm of feature selection algorithm to speed up it. Nordine Melab proposed one parallel wrapper feature selection based on genetic algorithm [4]. Hugo Silva el proposed one parallel feature selection based on genetic algorithm [5].

Based on the analysis above, this paper proposed one multi-population co-genetic algorithm for feature selection algorithm (MPAGAFS) inspired by our prevoious work[10].

[*] Corresponding author.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 493–500, 2009.

2 Introduction of Algorithm

2.1 Agent Structure

2.1.1 Closed Chain-Like Agent Structure Inside Subpopulation

In the structure, all the agents live in a close chain-like environment, L. The size of L is $1 \times L_{size}$, where L_{size} is an integer, 1 means one dimensional agent structure.

Definition: Assuming that the agent that is located at $(1, i)$ is represented as $L_{1,i}, i = 1, 2, \cdots, L_{size}$, the neighbors of $L_{1,i}$, $Neibors_{1,i}$ are defined as follows:

$$Neibors_{1,i} = \left\{ L_{1,i_1}, L_{1,i_2} \right\} \tag{1}$$

where $i_1 = \begin{cases} i-1 & i \neq 1 \\ L_{size} & i = 1 \end{cases}$, $i_2 = \begin{cases} i+1 & i \neq L_{size} \\ 1 & i = L_{size} \end{cases}$. In the closed chain-like agent structure, the agent can interact with the left and right neighborhood ones.

2.1.2 Multi-population Cycle Chain-Like Agent Structure

Multi-population cycle chain-like agent structure means that the whole population is divided into some subpopulations; the neighboring subpopulations have common agents which are shared agents. Figure 1 shows the agent structure with 6 agents per subpopulation and 2 shared agents. They cooperate and compete with each other. Finally, the agent with low power will die, and new agent will occupy its position. The introduction of the shared agents will supply the genetic information of other subpopulations. Through the sharing of the agents, the genetic information can spread from some subpopulation to the neighboring subpopulations.

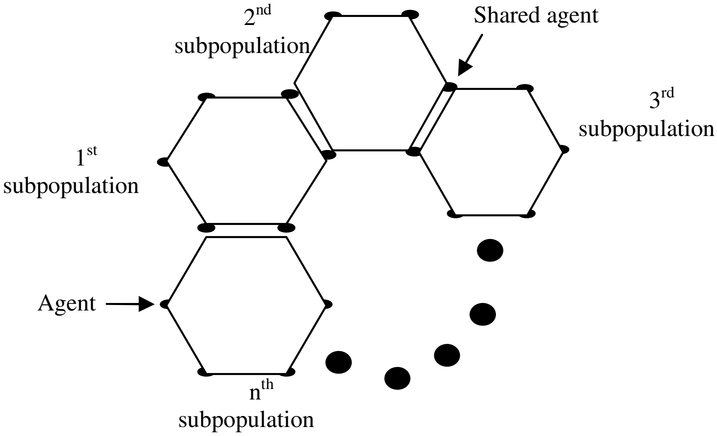


Fig. 1. Multi-population cycle chain-like agent structure

2.2 Genetic Operators

2.2.1 Dynamic Neighborhood Competition Selection Operator

Suppose the order of competition selection is from left to right, the current agent is $L^t_{1,i}$, the neighbors are $Nbs_{1,i}$, $Nbs_{1,i} = \left\{ L^t_{1,i1} \quad L^t_{1,i2} \right\}$, $i = 1, 2 \ldots\ldots popsize$. Updating of $L^t_{1,i}$ is as the following formula:

$$\left\{ \begin{array}{ll} L^t_{1,i} = L^t_{1,i} & fitness\left(L^t_{1,i}\right) > fitness\left(\max\left(L_{1,i1}, L_{1,i2}\right)\right) \\ L^t_{1,i} = L^t_{1,i} \circ L^t_{1,i1} & \max\left(L_{1,i1}, L_{1,i2}\right) = L_{1,i1} \ \& \ fitness\left(L_{1,i1}\right) > fitness\left(L^t_{1,i}\right) \\ L^t_{1,i} = L^t_{1,i} \circ L^t_{1,i2} & \max\left(L_{1,i1}, L_{1,i2}\right) = L_{1,i2} \ \& \ fitness\left(L_{1,i2}\right) > fitness\left(L^t_{1,i}\right) \end{array} \right\} \tag{2}$$

In the formula (2), $\circ$ means competition selection between agent $L^t_{1,i}$ and $L^t_{1,i1}$, the two agents consist of lots of genes:

$$\begin{array}{l} L^t_{1,i} = \left(c^t_{i,1} \quad c^t_{i,2} \quad \cdots \quad c^t_{i,j} \quad \cdots \quad c^t_{i,length}\right) \\ L^t_{1,i1} = \left(c^t_{i1,1} \quad c^t_{i1,2} \quad \cdots \quad c^t_{i1,j} \quad \cdots \quad c^t_{i1,length}\right) \end{array} \tag{3}$$

$c^t_{i,j}$ means jth gene of $L^t_{1,i}$, $c^t_{i1,j}$ means jth gene of $L^t_{1,i1}$, $length$ means number of genes of single agent. The $\circ$ processing is as follows:

$$\begin{cases} \mathrm{c}^t_{i,j} = \mathrm{c}^t_{i,j} & \mathrm{c}^t_{i,j} = \mathrm{c}^t_{i1,j} \\ \mathrm{c}^t_{i,j} = \mathrm{U}(0,1) & \mathrm{c}^t_{i,j} \neq \mathrm{c}^t_{i1,j} \end{cases} \tag{4}$$

$\mathrm{U}(0,1)$ means random number generator and is within the domain $[0,1]$.

2.2.2 Neighborhood Adaptive Crossover Operator

The crossover probability $p_{c,i}$ is calculated adaptively as formula (5):

$p_{c,i}$ means the probability of crossover between the $L_{1,i}$ and $Max_{1,i}$, $GH(i,i')$ means the distance between $L_{1,i}$ and $Max_{1,i}$, f' means the maximum value of all the individuals, $f_{\max}$ means the maximum value of all the individuals in the current population, f_{ave} mean the average fitness value of all the individuals.

$$p_{c,i} = \begin{cases} \left(\dfrac{f_{\max} - f'_i}{f_{\max} - f_{ave}}\right)^{\frac{1}{GH(i,i')}} & f' \geq f_{ave} \\ 1 & f' < f_{ave} \end{cases} \tag{5}$$

2.2.3 Adaptive Mutation Operator

The mutation probability p_m is calculated adaptively based on the length of chromosome: $p_m = \frac{1}{n}$, where n means number of genes (or features here).

2.3 Local Search Operator

The algorithm about local search operator is described as follows:

Step 1: $r \leftarrow 0$;
Step 2: do dynamic neighborhood competitive selection and obtaining $L^{r+1/2}$;
Step 3: for each agent in $L^{r+1/2}$, do mutation processing on it, obtaining L_{end}^{r};
Step 4: find ind_{best}^{cr} in L_{end}^{r}, if $Eng\left(ind_{best}^{cr}\right) > Eng\left(ind_{best}^{r-1}\right)$,
then $ind_{best}^{r} \leftarrow ind_{best}^{cr}$, else, $ind_{best}^{r} \leftarrow ind_{best}^{r-1}$, $L^{r+1} \leftarrow L_{end}^{r}$.
Step 5: if stop criterion is satisfied, then go the next generation of MPAGAFS_IN, else, $i \leftarrow i+1$, go to Step 2.

Comment: *L^r represents the agent chain in the ith generation of local searching within some generation, $L^{r+1/2}$ is the mid-chains between L^r and L^{r+1}, L_{end}^{r} is the agent chain after mutation processing in the ith generation of local searching within some generation. ind_{best}^{r} is the best agent since initialization of population and ind_{best}^{cr} is the best agent in L^r of local searching within some generation.*

2.4 Realization of MPAGAFS Algorithm

The MPAGAFS algorithm can be divided into two parts: MPAGAFSFS inside subpopulation (MPAGAFS_IN) and MPAGAFS between subpopulations (MPAGAFS_BETWEEN).

In MPAGAFS_BETWEEN, the major procedures are as follows:

Step 1: the initial population is obtained.
Step 2: the initial population is divided into M subpopulations with the size of L
Step 3: each subpopulation evolves respectively.
Step 4: judge whether all the subpopulations finish their one generation evolution, if so, the M best individuals are obtained; if not, continue step 3.
Step 5: The M best individuals can be judged and used to update the best individual in the whole population in the current generation.
Step 6: judge whether the stop criterion is satisfied, if so, output the final best individual; if not, turn to step 3.

In MPAGAFS_IN, The major procedures are as follows:

Step 1: initialize L^0, update pop_{best}^0, and $t \leftarrow 0$;

Step 2: do dynamic neighborhood competitive selection processing and update L^t, obtaining $L^{t+1/3}$;

Step 3: for each agent in $L^{t+1/3}$, do crossover processing on it, obtaining $L^{t+2/3}$;

Step 4: for each agent in $L^{t+2/3}$, do mutation processing on it, obtaining L_{end}^t;

Step 5: find ind_{best}^{ct} in L_{end}^t, if $Eng\left(ind_{best}^{ct}\right) > Eng\left(ind_{best}^{t-1}\right)$, then $ind_{best}^t \leftarrow ind_{best}^{ct}$, else, $ind_{best}^t \leftarrow ind_{best}^{t-1}$, $L^{t+1} \leftarrow L_{end}^t$.

Step 6: if stop criterion is satisfied, then output ind_{best}^t, else $t \leftarrow t+1$, go to Step 2.

Comment: *L^t represents the agent chain in the tth generation, $L^{t+1/3}$ and $L^{t+2/3}$ are the mid-chains between L^t and L^{t+1}, L_{end}^t is the agent chain after mutation processing in the tth generation. ind_{best}^t is the best agent since initialization of population, ind_{best}^{ct} is the best agent in L^t. p_c and p_m are the probabilities to perform the neighborhood crossover processing and the mutation processing.*

3 Experiments and Analysis of Results

From the agent structure of MPAGAFS, it is seen that the algorithm can realize multi-subpopulation parallel feature selection, so the time cost can be reduced a lot. However, whether the MPAGAFS can obtain the satisfying feature selection precision is not for sure. The feature selection precision is important. In order to show the feature selection precision of the MPAGAFS, groups of experiments are reported here.

In the following experiments, we set 6 as the size of subpopulation and 2 as the number of shared agents. The setup of other parameters is: the size of whole population is 66, the initial probability of crossover and mjutation are $p_c = 0.95$ and $P_m = 0.05$ respectively, the upper limit of evolution generation is $T = 1000$, TIMEs_OUT=10. The relevant condition about PC platform is: CPU with mainframe of 2.8GHz, memory of 0.99GB.

The fitness function for feature selection here: Fitness function (evaluation criterion): $fitness = \sum_{i=1}^{N} ({}^{S_b}\!/\!_{S_w})_i - corr2$, where, N : number of the features; S_b :between-classes variance, $S_b = \left(m_1 - m_2\right)^2$; m_1 : the first class specimens under some feature; m_2 : the second class specimens under the same feature; $S_w = \left(\sigma_{class1}\right)^2 + \left(\sigma_{class2}\right)^2$; $corr2$: correlation between features selected.

3.1 Feature Selection Experiments with Filter Methods

Four genetic algorithms including AGA [2], MAGA [6], SFGA [7] and SGAE [8] are adopted to be compared with MPAGAFS for feature selection. The table 1 shows some information. The datasets Letter, Wave and Sonar are from UCI database [9].

The table 2 lists average experimental results of the 10 times experiments. From the table 2, it can be seen the average number of features from MPAGAFS is less than that from MAGA and SFGA, the variance of the average number of features from

Table 1. Some information about datasets

Datasets	Number of features	Number of specimens	Number of classes	Population size	Evaluation of feature subset	Stop criterion
Letter	16	1555	2	30	5-fold CV	k>10 or 1000
Wave	40	2000	2	30	5-fold CV	k>10 or 1000
Sonar	60	208	2	30	5-fold CV	k>10 or 1000

*Here the stop criterion is k>10 or maximum number of iteration is more than 1000.

Table 2. Comparison of feature selection capability of five GAs

DS	CP	SGAE	AGA	SFGA	MAGA	MPAGAFS
Letter	ANF	10.2, ± 3.6	9.9, ± 2.7	10.5, ± 3.9	11, ± 2.8	10.2, ± 2.1
	ABF	17.6629, ± 1.47	17.6897, ± 0.23	17.9449, ± 0.34	17.7852, ± 0.12	17.9449, ± 0
	ACA	92.75, ± 3.4	95, ± 2.4	95, ± 4.8	93.5, ± 3.1	98, ± 1.7
Wave	ANF	18.4, ± 4.8	21.3, ± 4.3	18.4, ± 4.4	15.5, ± 3.8	14.9, ± 3.6
	ABF	-0.1269, ± 0.13	-0.5034, ± 0.15	0.4300, ± 0.11	1.5769, ± 0.09	1.5467, ± 0.06
	ACA	80.25, ± 4.8	74.25, ± 5.4	80.25, ± 4.8	77, ± 2.8	82.75, ± 2.6
Sonar	ANF	25.7, ± 3.4	25.8 ± 4.5	25.4 ± 4.7	25.8 ± 2.8	26.7 ± 2.4
	ABF	12.4563, ± 0.15	12.5634 ± 0.23	12.7773 ± 0.17	11.892 ± 0.19	13.4572 ± 0.14
	ACA	89.9	92.9	92.6	95.5	95.6

*ANF means average number of selected features, ABF means the average best fitness value, ACA means average classification accuracy of selected feature subset.

MPAGAFS is lowest. The average best fitness value from MPAGAFS is highest and most stable. The average classification accuracy from MPAGAFS is highest and the variance of the average classification accuracy is lowest.

3.2 Comparison with Parallel Feature Selection Method

Here, the parallel feature selection method [5] is compared with MPAGAFS. The relevant papers do not state how to divide the feature space in detail, so we list two kind of division; they correspond to two kinds of methods (method 1 and method 2). For method 1, it fixes the former several features to divide the feature space into several subspaces; here we suppose the number of fixed features is 3, so the number of feature subspace is $2^3=8$. For method 2, it fixes the latter several features to divide the feature space into several subspaces; here we suppose the number of fixed features is 3, so the number of feature subspace is $2^3=8$ too. All the three feature selection method use the same classifier (BP neural network) for wrapper feature selection.

Table 3. Comparison with parallel feature selection method

DS	CP	MPAGAFS (-BP)	Method 1 (-BP)	Method 2 (-BP)
Letter	ANF	10, ±2.7	12.1, ±4.3	13.3, ±4.8
	ACA	98.1, ±1.7	92.1, ±3.1	91.7, ±3.7
Wave	ANF	14.9, ±3.7	17.7, ±3.7	18.1, ±4.5
	ACA	89.3, ±2.4	73.2, ±5.2	78.2, ±4.8
Sonar	ANF	24.2, ±3.4	28.9, ±4.4	27.6, ±5.6
	ACA	97.7, ±2.7	89.1, ±4.3	89.8 ±4.7

The table 3 lists the experimental results. It shows that this algorithm MPAGAFS is better than other methods in terms of number of features and classification accuracy. According to the datasets, the MPAGAFS can obtain least features, and lowest variance. Besides, it can obtain best classification accuracy and lowest variance.

4 Conclusions

In this paper, the authors propose one novel agent genetic algorithm-multi-population agent genetic algorithm (MPAGAFS) to enhance the precision and time cost of feature selection. The experimental results show that the MPAGAFS can obtain more precise and stable feature selection precision than four other popular GAs. Future works are to apply the MPAGAFS into data mining, decision making, and so on.

Acknowledgement

This research is funded by NSFC (No: 30570473), CSTC: 2008BB2164 and Innovation Ability Training Foundation of Chongqing University (CDCX018).

References

1. Leung, Y.W., Wang, Y.: An orthogonal genetic algorithm with quantization for global numerical optimization. IEEE Transaction on evolutionary computation 5(1), 41–53 (2001)
2. Srinivas, M., Patnaik, L.M.: Adaptive probabilities of crossover genetic in mutation and algorithms. IEEE Transactions on systems, man and cybernetics 24(4), 656–667 (1994)
3. Oh, I.-S., Lee, J.-S., Moon, B.-R.: Hybrid genetic algorithms for feature selection. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(11), 1424–1437 (2004)
4. Melab, N., Cahon, S., Talbi, E.-G., Duponchel, L.: Parallel GA-based wrapper feature selection for spectroscopic data mining. In: Proceedings of the International Parallel and Distributed Processing Symposium, 2002, pp. 201–208. IEEE Press, Los Alamitos (2002)
5. Silva, H., Fred, A.: Feature subspace ensembles: A parallel classifier combination scheme using feature selection. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 261–270. Springer, Heidelberg (2007)
6. Zhong, W., Liu, J., Xue, M., Jiao, L.: A multiagent genetic algorithm for global numerical optimization. IEEE transactions on systems,man, and cybernetics-part B:cybernetics 34(2), 1128–1141 (2004)
7. Gong, D.-W., Sun, X.-Y., Guo, X.-J.: Novel survival of the fittest genetic algorithm. Control and decision 17(6), 908–911 (2002)
8. Michalewicz, Z., Fogel, D.B.: How to solve it: modern heuristics, pp. 83–234. Springer, Heidelberg (2000)
9. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI repository of machine learning databases, Irvine, CA (1998), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
10. Zeng, X.-P., Li, Y.-M., Qin, J.: A Dynamic chain-like agent genetic algorithm for global numerical optimization and feature selection. Neurocomputing 72(4), 1214–1228 (2008)

Particle Swarm Optimization and Differential Evolution in Fuzzy Clustering

Fengqin Yang[1], Changhai Zhang[1], and Tieli Sun[2,⋆]

[1] College of Computer Science and Technology, Jilin University,
Changchun, Jilin, 130012, China
[2] College of Computer Science, Northeast Normal University,
Changchun, Jilin, 130117, China
suntl@nenu.edu.cn

Abstract. Fuzzy clustering helps to find natural vague boundaries in data. The fuzzy c-means (FCM) is one of the most popular clustering methods based on minimization of a criterion function because it works fast in most situations. However, it is sensitive to initialization and is easily trapped in local optima. Particle swarm optimization (PSO) and differential evolution (DE) are two promising algorithms for numerical optimization. Two hybrid data clustering algorithms based the two evolution algorithms and the FCM algorithm, called HPSOFCM and HDEFCM respectively, are proposed in this research. The hybrid clustering algorithms make full use of the merits of the evolutionary algorithms and the FCM algorithm. The performances of the HPSOFCM algorithm and the HDEFCM algorithm are compared with those of the FCM algorithm on six data sets. Experimental results indicate the HPSOFCM algorithm and the HDEFCM algorithm can help the FCM algorithm escape from local optima.

1 Introduction

Fuzzy clustering algorithms aim to model fuzzy (i.e., ambiguous) unsupervised (unlabeled) patterns efficiently, and one widely used algorithm is the fuzzy c-means (FCM) algorithm[1]. The FCM algorithm is based on an iterative optimization of a fuzzy objective function. Due to its efficacy, simplicity and computational efficiency, it is a very popular technique. However, the main drawback of the FCM algorithm is that the cluster result is sensitive to the selection of the initial cluster centers and may converge to the local optima.

In order to solve the problem of local optima encountered in the FCM algorithm, one possibility is to use some stochastic search methods. Recently, many fuzzy clustering algorithms based on evolutionary algorithms have been introduced. For instance, Klawonn and Keller[2] optimized the FCM model with a genetic algorithm. The particle swarm optimization(PSO) algorithm was applied to

[⋆] Corresponding author.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 501–508, 2009.

cluster suppliers under fuzzy environments into manageable smaller groups with similar characteristics[3]. Das et al. used an improved differential evolution(DE) algorithm to automatically determine the number of naturally occurring clusters in the image as well as to refine the cluster centers[4].

Experimental studies have shown that evolutionary algorithms improve the performance of the FCM algorithm[2][3][4], but at the same time there are some shortcomings such as slow convergence speed. The main goal of our study is to investigate the way combining evolutionary algorithms with the FCM algorithm to form hybrid fuzzy clustering algorithms. We combine PSO and FCM to form the hybrid clustering algorithm HPSOFCM, and combine DE and FCM to form the hybrid clustering algorithm HDEFCM. The experimental results on a variety of data sets provided from six artificial and real-life situations indicate the HPSOFCM algorithm and the HDEFCM algorithm can help the FCM algorithm escape from local optima.

2 The FCM Algorithm

In contrast to crisp clustering methods, which allocate each object to a unique cluster, fuzzy clustering algorithms result in membership values between 0 and 1 that indicate the degree of membership for each object to each of the clusters. In the fuzzy clustering algorithms, the most popular is the FCM algorithm. The FCM algorithm to be focused on here is based on minimizing the criterion[5][6]

$$J = \sum_{j=1}^{C} \sum_{i=1}^{N} \mu_{ij}^{m} d_{ij}^{2},\ m \geq 1\ . \tag{1}$$

with respect to the membership values μ_{ij} and the distance d_{ij}. The membership values μ_{ij} satisfy the stochastic constraints:

$$\mu_{ij} \in [0,1],\ 1 \leq j \leq C,\ 1 \leq i \leq N\ . \tag{2}$$

$$\sum_{j=1}^{C} \mu_{ij} = 1,\ \forall i = 1, \cdots, N\ . \tag{3}$$

$$0 < \sum_{i=1}^{N} \mu_{ij} < N, \forall j = 1, \cdots, C\ . \tag{4}$$

Here N is the number of objects and C is the number of clusters. m is often called the fuzzifier parameter and determines the fuzziness of the clustering. The clustering becomes fuzzier for larger values of m. Usually m is set to equal to 2, as this value has been proven to give good results with FCM. In the ordinary FCM algorithm, the distance between object i and cluster j, d_{ij}, is defined as the Euclidean distance between $\mathbf{x}_i$ and $\mathbf{v}_j$ where $\mathbf{x}_i$ denotes the data vector for object $i(i = 1, \cdots, N)$ and $\mathbf{v}_j$ denotes the prototype vector for cluster $j(j = 1, \cdots, C)$.

Applying derivative to Eqs. (1), (2), (3) and (4), one can derive the computational formulae of μ_{ij} and $\mathbf{v}_j$ as:

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C}(d_{ij}/d_{ik})^{2/(m-1)}},\ 1 \le j \le C,\ 1 \le i \le N, m > 1\ . \tag{5}$$

$$\mathbf{v}_j = \frac{\sum_{i=1}^{N}(\mu_{ij})^m \mathbf{x}_i}{\sum_{i=1}^{N}(\mu_{ij})^m},\ 1 \le j \le C\ . \tag{6}$$

The termination criterion for the FCM algorithm is usually chosen as max_{ik} $(|\mu_{ik}^{(l)} - \mu_{ik}^{(l-1)}|) < \epsilon$, where l denotes the number of iterations. The FCM algorithm converges to a local minimum of the c-means functional(1). Hence, different initializations may lead to different results.

3 PSO and DE

3.1 PSO

PSO is a population based stochastic optimization technique inspired by the social behavior of bird flock(and fish school, etc.), as developed by Kennedy and Eberhart[7]. As a relatively new evolutionary paradigm, it has grown in the past decade and many studies related to PSO have been published[8]. In PSO, each particle is an individual, and the swarm is composed of particles. The problem solution space is formulated as a search space. Each position in the search space is a correlated solution of the problem. Particles cooperate to find the best position(best solution) in the search space(solution space). Each particle moves according to its velocity. In each iteration, the particle movement is computed as follows:

$$x_i(t+1) = x_i(t) + v_i(t)\ . \tag{7}$$

$$v_i(t+1) = \omega v_i(t) + c_1 r_1 (pbest_i(t) - x_i(t)) + c_2 r_2 (gbest(t) - x_i(t))\ . \tag{8}$$

In Eqs.(7) and (8), $x_i(t)$ is the position of particle i at time t, $v_i(t)$ is the velocity of particle i at time t, $pbest_i(t)$ is the best position found by particle i itself so far, $gbest(t)$ is the best position found by the whole swarm so far, ω is an inertia weight scaling the previous time step velocity, c_1 and c_2 are two acceleration coefficients that scale the influence of the best personal position of the particle $(pbest_i(t))$ and the best global position $(gbest(t))$, r_1 and r_2 are random variables between 0 and 1.

3.2 DE

DE was first introduced by Price and Storn in 1995[9] and has been drawing an increasing attention recently[10][11]. DE is a population based global optimization algorithm that uses floating-point encoding and combines simple arithmetic operators with the classical events of mutation, crossover and selection to evolve from a randomly generated initial population to a satisfactory one. DE starts with the random initialization of a population of individuals in the search space and creates a competitor for each individual by mutation and crossover. The mutation is completed by the following formulation:

$$\widetilde{\mathbf{x}}_i = \mathbf{x}_1 + F(\mathbf{x}_2 - \mathbf{x}_3) \ . \tag{9}$$

where $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ are three different individuals of the population, F is a parameter between [0,1], and $\mathbf{x}_2 - \mathbf{x}_3$ denotes the differential item. Then a trial vector $\mathbf{y}_i$ as a competition to target point $\mathbf{x}_i$ will be found from its parents $\mathbf{x}_i$ and $\widetilde{\mathbf{x}}_i$ using the following crossover rules:

$$\mathbf{y}_i^j = \begin{cases} \widetilde{\mathbf{x}}_i^j & if\ R^j \leq C_R\ or\ j = I_i \\ \mathbf{x}_i^j & if\ R^j > C_R\ and\ j \neq I_i \end{cases} \ . \tag{10}$$

where I_i is a randomly chosen integer,i.e. $I_i \in \{1, 2, \cdots, n\}$, where n is the dimension of the vector; the superscript j represents the jth component of the respective vector; $R_j \in (0, 1)$, draw randomly for each j. The entity C_R is a constant. Then the population is updated by the following formulation:

$$\mathbf{x}_i(g+1) = \begin{cases} \mathbf{y}_i(g) & if\ \mathbf{y}_i(g)\ is\ better\ than\ \mathbf{x}_i(g) \\ \mathbf{x}_i(g) & otherwise \end{cases} \ . \tag{11}$$

where g denotes the number of generations.

4 The Proposed Hybrid Clustering Algorithms

4.1 Encoding Mechanism

In the hybrid clustering algorithm, an individual represents the center points of the clusters and is encoded into a string of real numbers. The real number string has the form shown in Fig.1, where n is the dimension of the data to be clustered, C is the number of clusters.

x_{11}	$\cdots$	x_{1i}	$\cdots$	x_{1n}	$\cdots$	x_{j1}	$\cdots$	x_{ji}	$\cdots$	x_{jn}	$\cdots$	x_{C1}	$\cdots$	x_{Ci}	$\cdots$	x_{Cn}

Fig. 1. The representation of an individual

```
1.Set the initial parameters including the maximum iterative count $IterCount$,
$PsoCount$, $FcmCount$, the population size $P_{size}$, $\omega$, $c_1$ and $c_2$.
2.Initialize a population of size $P_{size}$.
3.Set iterative count $Gen_1 = 0$.
4.Set iterative count $Gen_2 = Gen_3 = 0$.
5.PSO Method
  (1)Update the $P_{size}$ particles according Eqs. (7) and (8).
  (2)$Gen_2 = Gen_2 + 1$. If $Gen_2 < PsoCount$, go to 5(1).
6.FCM Method
  (1)Take the position of each particle $i$ as the initial cluster center vector.
  (2)Recalculate each cluster center vector using Eqs. (5) and (6).
  (3)$Gen_3 = Gen_3 + 1$. If $Gen_3 < FcmCount$, go to 6(2).
7.$Gen_1 = Gen_1 + 1$. If $Gen_1 < IterCount$, go to 4, otherwise terminate.
```

Fig. 2. The HPSOFCM algorithm

```
1.Set the initial parameters including the maximum iterative count $IterCount$,
$DeCount$, $FcmCount$, the population size $P_{size}$, $F$ and $C_R$.
2.Initialize a population of size $P_{size}$.
3.Set iterative count $Gen_1 = 0$.
4.Set iterative count $Gen_2 = Gen_3 = 0$.
5.DE Method
  (1)Updating the $P_{size}$ individuals according Eqs. (9), (10) and (11).
  (2)$Gen_2 = Gen_2 + 1$. If $Gen_2 < DeCount$, go to 5(1).
6.FCM Method
  (1)Take each individual $i$ as the initial cluster center vector.
  (2) Recalculate each cluster center vector using Eqs. (5) and (6).
  (3) $Gen_3 = Gen_3 + 1$. If $Gen_3 < FcmCount$, go to 6(2).
7.$Gen_1 = Gen_1 + 1$. If $Gen_1 < IterCount$, go to 4, otherwise terminate.
```

Fig. 3. The HDEFCM algorithm

4.2 The Proposed Hybrid Clustering Algorithms

The FCM algorithm tends to converge faster than the evolutionary algorithms because it requires fewer function evaluations, but it usually gets stuck in local optima. We integrate the FCM algorithm with the evolutionary algorithms to form the hybrid clustering algorithms, which maintain the merits of the both kinds of algorithms. More specifically, the hybrid clustering algorithms will apply the FCM algorithm with m iterations to the individuals in the population every l generations such that the fitness value of each individual is improved. We combine PSO and FCM to form the hybrid clustering algorithm HPSOFCM, and combine DE and FCM to form the hybrid clustering algorithm HDEFCM. Fig. 2 and Fig. 3 outline the HPSOFCM algorithm and the HDEFCM algorithm respectively. The objective function of the FCM algorithm J defined in Eq.(1) is the fitness functions of the hybrid clustering algorithms.

5 Experimental Results

5.1 Data Sets

Six data sets are employed to validate our methods. These data sets, named ArtSet1, ArtSet2, Iris, breast-cancer-wisconsin (denoted as Cancer), Contraceptive Method Choice (denoted as CMC) and Wine, cover examples of data of low, medium and high dimensions. All data sets except ArtSet1 and ArtSet2 are available at ftp://ftp.ics.uci.edu./pub/machine-learning-databases/. Table 1 summarizes the characteristics of these data sets. ArtSet1 and ArtSet2 are artificial data sets. ArtSet1 is a two-featured problem with three unique classes. The patterns are drawn from three independent bivariate normal distributions, where classes are distributed according to $N_2(\mu_i = \begin{pmatrix} \mu_{i1} \\ \mu_{i2} \end{pmatrix}, \Sigma = \begin{pmatrix} 1 & 0.04 \\ 0.04 & 1 \end{pmatrix})$, $\mu_{11} = \mu_{12} = -2$, $\mu_{21} = \mu_{22} = 1$, $\mu_{31} = \mu_{32} = 4$, μ and Σ being mean vector and covariance matrix respectively. ArtSet2 is a three-featured problem with three classes, where every feature of the classes is distributed according to $Class1 \sim Uniform(5, 10)$, $Class2 \sim Uniform(10, 15)$, $Class3 \sim Uniform(15, 20)$.

5.2 Experimental Results

We evaluate and compare the performances of FCM, HPSOFCM and HDEFCM in terms of the fitness value, the Xie-Beni index[12] and the runtime. In our experiments, $m = 2$ and $\epsilon = 0.001$. The parameters shown in Table 2 are set. Table 3 gives the means, the best values (in brackets) and standard deviations (in square brackets) over 10 runs obtained for each of these measures. Bold face and italic face indicate the best and the second best results out of the three algorithms respectively. We can see that FCM is fastest, but it can not get the best Xie-Beni index or the best fitness values. HDEFCM can always get the best Xie-Beni index and better fitness values. HPSOFCM can always obtain the best fitness values and is faster than HDEFCM. That is to say HPSOFCM and HDEFCM improve the performance of FCM in terms of the fitness value and the Xie-Beni index respectively.

Table 1. Characteristics of the data sets

Name of data set	No. of classes	No. of features	Size of data set (size of classes)
ArtSet1	3	2	300(100,100,100)
ArtSet2	3	3	300(100,100,100)
Iris	3	4	150(50,50,50)
Cancer	2	9	683(444,239)
CMC	3	9	1473(629,334,510)
Wine	3	13	178(59,71,48)

Table 2. The parameters setup

HPSOFCM	HDEFCM
$IterCount$ 5	$IterCount$ 5
$FcmCount$ 4	$FcmCount$ 4
$PsoCount$ 4	$DeCount$ 4
P_{size} 20	P_{size} 20
ω 0.7298	F 0.5
c_1 1.49618	C_R 0.5
c_2 1.49618	

Table 3. Experiment results of FCM, HDEFCM, and HPSOFCM clustering on two artificial and four real data sets The quality of clustering is evaluated using the fitness value, the Xie-Beni index and the runtime. The table shows means ,the best value(in brackets) and standard deviations (in square brackets) for 10 independent runs. Bold face indicates the best and italic face indicates the second best result out of the three algorithms.

ArtSet1	FCM	HDEFCM	HPSOFCM
ObjValue	490.94(490.94)	*490.75(490.73)*	**490.73(490.72)**
	[0.003]	*[0.011]*	**[0.007]**
Xie-Beni	*8.9066(8.9063)*	**8.9033(8.9028)**	9.0095(8.9810)
	[0.000]	**[0.000]**	[0.015]
Runtime	**0.0152(0.0150)**	6.6022(6.5946)	*1.5060(1.500)*
	[0.000]	[0.005]	*[0.008]*
ArtSet2	FCM	HDEFCM	HPSOFCM
ObjValue	1752.61(1752.60)	*1752.54(1752.50)*	**1752.48(1752.46)**
	[0.001]	*[0.024]*	**[0.014]**
Xie-Beni	*7.8882(7.8882)*	**7.8879(7.8878)**	7.9055(7.8919)
	[0.000]	**[0.000]**	[0.010]
Runtime	**0.0150(0.0150)**	9.4589(9.4370)	*1.6451(1.6400)*
	[0.000]	[0.015]	*[0.008]*
Iris	FCM	HDEFCM	HPSOFCM
ObjValue	67.82(67.82)	*67.72(67.58)*	**67.53(67.50)**
	[0.002]	*[0.068]*	**[0.028]**
Xie-Beni	*6.6182(6.6180)*	**6.6070(6.5947)**	7.1171(6.7448)
	[0.000]	**[0.007]**	[0.173]
Runtime	**0.0166(0.0150)**	4.8855(4.8750)	*0.8870(0.8750)*
	[0.005]	[0.007]	*[0.006]*
Cancer	FCM	HDEFCM	HPSOFCM
ObjValue	17190.42(17190.34)	*17187.81(17184.82)*	**12289.61(11806.74)**
	[0.039]	*[1.767]*	**[297.147]**
Xie-Beni	*42.0172(42.0164)*	**42.0093(42.0021)**	162.7880(153.8223)
	[0.000]	**[0.004]**	[4.640]
Runtime	**0.0325(0.0310)**]	13.3952(13.3590)	*5.4077(5.3910)*
	[0.005]	[0.037]	*[0.015]*
CMC	FCM	HDEFCM	HPSOFCM
ObjValue	19588.44(19588.26)	*19586.11(19583.92)*	**19582.87(19582.06)**
	[0.265]	*[1.144]*	**[0.363]**
Xie-Beni	*65.3895(65.3895)*	**65.3865(65.3509)**	67.0668(66.6713)
	[0.001]	**[0.027]**	[0.378]
Runtime	**0.2812(0.2030)**	53.5266(53.4690)	*11.7093(11.7030)*
	[0.039]	[0.039]	*[0.008]*
Wine	FCM	HDEFCM	HPSOFCM
ObjValue	709190.74(494465.19)	*494390.23(494380.67)*	**494390.07(494379.36)**
	[452664.024]	*[7.210]*	**[5.426]**
Xie-Beni	9.8048(0.8841)	**0.8840(0.8840)**	*0.8858(0.8841)*
	[18.807]	**[0.000]**	*[0.002]*
Runtime	**0.0277(0.0150)**	7.0994(7.0930)	*1.7467(1.7340)*
	[0.010]	[0.008]	*[0.010]*

6 Conclusion

This paper investigates the hybrid clustering algorithms based on the FCM algorithm and the evolutionary algorithms. We combine PSO and FCM to form the hybrid clustering algorithm HPSOFCM, and combine DE and FCM to form the hybrid clustering algorithm HDEFCM. The experimental results indicate the HPSOFCM algorithm and the HDEFCM algorithm can help the FCM algorithm escape from local optima.

Acknowledgments. I would like to express my thanks and deepest appreciation to Prof. Jigui Sun. This work is partially supported by Science Foundation for Young Teachers of Northeast Normal University (No.20061006), Specialized Research Fund for the Doctoral Program of Higher Education (No.20050183065, No.20070183057).

References

1. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York (1981)
2. Klawonn, F., Keller, A.: Fuzzy Clustering with Evolutionary Algorithms. Int.J. of Intelligent Systems. 13(10-11), 975–991 (1998)
3. Mehdizadeh, E., Tavakkoli-Moghaddam, R.: A Hybrid Fuzzy Clustering PSO Algorithm for a Clustering Supplier Problem. In: IEEE International Conference on Industrial Engineering and Engineering Management, pp. 1466–1470 (2007)
4. Das, S., Konar, A., Chakraborty, U.K.: Automatic Fuzzy Segmentation of Images with Differential Evolution. In: IEEE Congress on Evolutionary Computation, pp. 2026–2033 (2006)
5. Berget, I., Mevik, B., Næs, T.: New Modiffcations and Applications of Fuzzy C-means Methodology. Computational Statistics and Data Analysis 52, 2403–2418 (2008)
6. Wang, X., Wang, Y., Wang, L.: Improving Fuzzy C-means Clustering Based on Feature-weight Learning. Pattern Recognition Letters 25, 1123–1132 (2004)
7. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of the IEEE intenational conference on neural networks, pp. 1942–1948. IEEE Press, New Jersey (1995)
8. Ali, M.M., Kaelo, P.: Improved Particle Swarm Algorithms for Global Optimization. Applied Mathematics and Computation 196, 578–593 (2008)
9. Storn, R., Price, K.: Differential Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical Report TR-95-012, University of California. ICSI, Berkeley (1995)
10. Huang, F., Wang, L., He, Q.: An Effective Co-evolutionary Differential Evolution for Constrained Optimization. Applied Mathematics and Computation 186, 340–356 (2007)
11. Qian, W., Li, A.: Adaptive Differential Evolution Algorithm for Multiobjective Optimization Problems. Applied Mathematics and Computation 201, 431–440 (2008)
12. Xie, X.L., Beni, G.: A Validity Measure for Fuzzy Clustering. IEEE Trans. Pattern Anal. Mach. Intell. 13(8), 841–847 (1991)

Intelligent Control of Heating, Ventilating and Air Conditioning Systems

Patrick Low Tiong Kie[1] and Lau Bee Theng[2]

[1] Kuching, Sarawak, Malaysia
[2]Swinburne University of Technology Sarawak Campus
School of Computing and Design, Sarawak, Malaysia
blau@swinburne.edu.my

Abstract. This paper proposed a simulation-optimization energy saving strategy for heating, ventilating and air conditioning (HVAC) systems' condenser water loop through intelligent control of single speed cooling towers' components. An analysis of system components has showed the interactions of control variables inside the cooling towers and between the cooling tower and chillers. Based on the analysis, a model based optimization approach was developed with evolutionary computation. A simulation application demonstrated the effectiveness of the proposed strategy. This strategy can also be easily modified and applied to single speed tools in the refrigerant loops.

1 Introduction

In modern HVAC systems, building cooling and heating sources consume the highest volume of the electricity. Consequently, the optimization of the building cooling and heating sources has been extensively studied in HVAC systems for more efficient part load operation using Variable Speed / Frequency drive (VSD/VFD). Global optimal control methods for plant cooling have been carried out by Sud [13], Lau et al. [9] and Johnson [8]. However those models are not for real time control. Braun et al. [2, 3, 4] has developed a strategy for optimal control of chilled water systems which was suitable for real time application. Yao et al. [15] has developed a mathematical model for optimizing cooling systems' operation based on energy analysis of the main dynamic facilities. On the other hand, Lu et al. [10] developed a global optimization strategy for heating, ventilating and air conditioning systems. However, the models do not consider the time dependent characteristics of parameters. Chang [5] presented an approach using Lagrangian method to solve the optimizing chiller loading. Chow et al. [6] introduced a concept of integrating artificial neural network and genetic algorithm optimization of absorption chillers. Fong et al. [7] introduced the evolutionary programming for optimizing chillers in the HVAC systems. Alcala et al. [1] developed a weighted linguistic fuzzy rules combined with a rule selection process for intelligent control of heating, ventilating and air conditioning systems concerning energy performance and indoor comfort requirements. Basically, most of the researches focus on chillers. The researches on HVAC systems discussed mainly on the newer systems in manufacturing plants that come with VSD to optimize the chiller water pumps, condenser water pumps, chillers and cooling towers [16]. The cost

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 509–516, 2009.

savings from VSD is very encouraging [14]. VFD is also used in dynamic Transient Simulation Program based simulation platform for alternative control strategies including set-point control logics of the supply cooling water temperature and cooling tower fan modulation control methods as well as different number control means of cooling towers [4]. However, upgrading the older HVAC systems which use single speed controllers requires huge costs (approx. USD30K for complete installation of single VSD/VFD) depending on the number and type of drives. Hence our proposed approach focuses on energy savings on these single speed HVAC systems through the computational intelligence. Energy saving on HVAC through computational intelligence optimization of cooling towers is feasible [10, 12]. Our main concern in modeling minimal power consumption for the whole condenser water loop is the single speed cooling tower fans and condenser water pumps.

2 Our Optimization Target: Condenser Water Loop

The main purpose of cooling towers is to supply condenser water to chillers using a condenser water loop. This is because the performance of a chiller is influenced by chilled water supply temperature (T_{shws}), condenser water supply temperature (T_{cws}) and cooling load (C). For cooling towers, the power consumptions of pumps and fans are influenced by two parameters. mass flow rates of water and the pressure difference between the inlets and outlets. The characteristics of pumps and fans are very similar. The plant's HVAC systems are simplified as shown in Fig. 1 with labels for components and measured readings. The main components are six cooling towers fans and six condenser water pumps servicing the whole chiller plant. The adjustment of single speed cooling tower fans and condenser water pumps has effects on the total water side heat load of cooling towers, H. Consequently, the heat load is affected by the mass flow and heat of water, condenser water temperature at inlet and outlet. Even though we aim to minimize the power consumption of cooling towers, the manipulation of the fans and pumps must be optimized to meet the cooling load of the chillers in the plant, C. The cooling load is measured from mass flow of chilled water, heat of water, chilled water supply and return temperatures. The chillers have two types of capacity, namely low temp and high temp. There are three low temp chillers which have a rated power of 1000 kW each and three high temp chillers which have a rated power of 750 kW each. The chillers and their chilled water pumps are controlled by variable speed drives. The total power consumption for chiller side is 5700 kW per hour. On the other hand, the estimation of cooling load (Equation 1), C [12] of the chillers is 3415.5 tons when they are fully operated. m_{chw} is the mass flow of chilled water, c_p is the heat of water under constant pressure, h_p is pounds of heat per galloon of water, T_{chwr} and T_{chws} are the chilled water return and supply temperatures. The chillers are set to perform at 80% of its designated capacity presently. Presently, the coefficient of performance, COP [12] of the low temp chiller is 2.00 and high temp chiller is 3.20 calculated using Equation 2. E_c represents energy contributed in Btu/h and P_a is the power required in watts. However, the total power consumption for the six cooling towers which consist of six single speed fans operating together with six single speed condenser water pumps are 715.2 kW per hour. Due to the mass flow of water at inlet ($m_{w,i}$) and outlet ($m_{w,o}$) has the same volume and makeup water (m_m) is only 0.001% of the mass flow, Braun [2] model (ε_a) in Equation 3 is simplified. The heat transfer effectiveness, μ based on Braun's model is summarized in Equation 4. It utilizes the mass flow and temperature of water at both inlet and outlet. Ambient temperature is measured by wet bulb at inlet i.e. 82.4^{O}F. Hence the designated efficiency of the cooling tower based on Braun's heat transfer effectiveness, μ is 0.6111. The cooling towers have the designated heat load capacity; *H* [11] is 8468.064 tons when they are fully operated. m_w is the mass flow of condenser water, T_{cwr} and T_{cws} are the condenser water return and supply temperatures is shown in Equation 5.

$$C = \frac{c_p h_p m_{chw} (T_{chwr} - T_{chws})}{12000} \tag{1}$$

$$COP = (E_c / P_a) / 3.412 \tag{2}$$

$$\varepsilon_a = \frac{m_{w,i} C_p T_{cwr} + m_m C_p T_m - m_{w,o} C_p T_{cws}}{m_{w,o} C_p (T_{cwr} - T_{wb})} \tag{3}$$

$$\mu = (T_{cwr} - T_{cws}) / (T_{cwr} - T_{wb}) \tag{4}$$

$$H = \frac{c_p h_p m_w (T_{cwr} - T_{cws})}{15000} \tag{5}$$

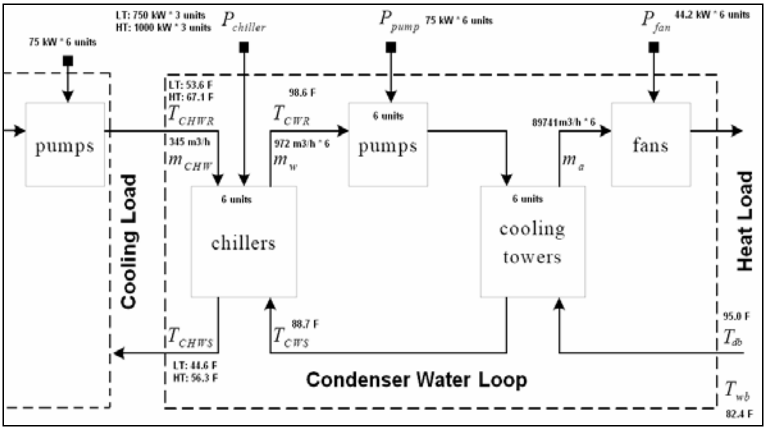


Fig. 1. Schematic view of condenser water loop

3 Objective Function

Looking into the costs of investing in another dozen of VSD, and also the heat load, we probe into the possibility of modeling the single speed fans and condenser water pumps to meet the heat load and cooling load on real time basis. Experiments have been conducted to find out the possible hazards of manipulating the single speed fans and pumps. Hence it is feasible as it does not trip or interrupt other equipments in the plant. The objective function is to minimize total power consumption of the condenser water loop where P_{CT} is the measured total power consumption of cooling tower consists of condenser water pumps and cooling tower fans, P_{CH} is the measured total power consumption of chillers and chilled water pumps in Equation 6. The chillers have variable speed drives that control the individual cooling load based on the capacity, adjustment factor for part load and temperature as shown in the equation below. Hence the total power consumption of the chillers can be obtained directly from the controller. Due to the use of VSD, the chiller efficiency has been maximized based on various condenser water supply, chilled water supply and return temperatures. P_{CH} [11] is measured by $Q_{cap,i}$, nominal capacity, PLR_i , part load factor and T_i , temperature factor of chiller i as in Equation 7. The modeling of cooling tower power consumption, P_{CT} is determined by total power consumption by single speed pumps and fans, P_{pump} and P_{fan} as in Equation 8. The power consumption of condenser water pumps are calculated based on the rated power, measured and nominal mass flow of condenser water flow. Since it is a single speed pump, hence the power consumption is zero when it is turned off. a_p is the pump mode, pm_0 is the rated power, $m_{w,p}$ measures the mass flow of water and $m_{wn,p}$ measures the nominal mass flow of water of *i-th* pump as calculated in Equation 9. As for the power consumption of single speed cooling tower fan, the nominal and measured mass flow of air and rated power are calculated (Equation 10). Hence, if the fan is turned off, the power consumption is zero indicated by fan mode, b_p=0. fn_0 is the rated power, $m_{a,p}$ measures the mass flow of air and $m_{an,p}$ measures the nominal mass flow of air of *i-th* pump.

$$P_{\min} = P_{CT} + P_{CH} \tag{6}$$

$$P_{CH} = \sum_{i=1}^{i\le 6} Q_{cap,i} COP \cdot PLR_i \cdot T_i \tag{7}$$

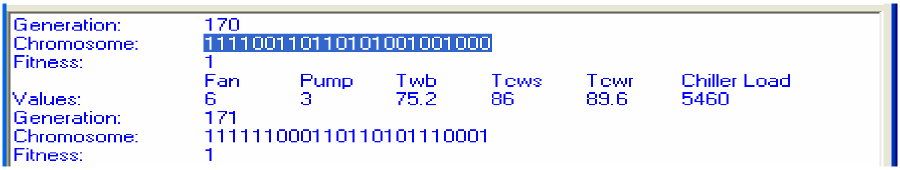
$$P_{CT} = P_{pump} + P_{fan} \tag{8}$$

$$P_{pump} = \sum_{p=1}^{p\le 6} a_p \frac{m_{w,p}}{m_{wn,p}} pm_o \tag{9}$$

$$P_{fan} = \sum_{f=1}^{f\le 6} b_p \frac{m_{a,p}}{m_{an,p}} fn_o \tag{10}$$

4 String Encoding

In order to meet our objective function, we model a set of related variables for optimization into a string which includes $(a_p, b_p, T_{wb}, T_{cws}, T_{cwr}, C)$ in Fig. 2. An initial population of random bit strings is obtained from the measured field data. The field data of this initial population is evaluated for their fitness or goodness in solving the problem. The initial population is evaluated to minimize P_{min} over the range of minimum and maximum values discussed. As P_{min} is nonnegative over the range, so it is used as the fitness of the string encoding.



```
Generation:     170
Chromosome:     111100110110101001001000
Fitness:        1
                Fan      Pump     Twb      Tcws     Tcwr     Chiller Load
Values:         6        3        75.2     86       89.6     5460
Generation:     171
Chromosome:     111111000110110101110001
Fitness:        1
```

Fig. 2. Strings and chromosome

5 Fitness Function

The fitness of a chromosome is evaluated based on the setting and fulfillment of the constraints. The fitness function, *f* is expressed in the following equation with penalties Pe_1 and Pe_2. Pe_1 assesses the chillers' cooling load, *C*. Pe_2 assesses the actual heat rejection capacity of cooling towers under the measured wet bulb, condenser water return and supply temperatures. The higher the fitness, *f* would signal the better the generation is. There are a few constraints that must be in place to validate the fitness function. All the variables must fall within the minimum and maximum allowed values.

$$f = \frac{1}{P_{\min} - (Pe_2 - Pe_1)} \tag{11}$$

$$Pe_1 = C \text{ And } Pe_2 = H \cdot (\frac{T_{cwr} - T_{cws}}{T_{cwr} - T_{wb}}) \tag{12}$$

$$H < C \text{ Where } H_{\min} \le H \le H_{\max} \text{ and } C_{\min} \le C \le C_{\max} \tag{13}$$

$$T_{cwr,\min} \le T_{cwr} \le T_{cwr,\max} \;\; T_{cws,\min} \le T_{cws} \le T_{cws,\max} ;\; T_{cws,\min} \le T_{cws} \le T_{cws,\max} ; \tag{14}$$

$$m_{w,\min} \le m_w \le m_{w,\max} \text{ and } m_{a,\min} \le m_a \le m_{a,\max} \tag{15}$$

Constraint 1: The heat load is measured in ton which has a minimum of 1800 ton to a maximum of 10800 from the six cooling towers. On the hand, the cooling load of six chillers measured in tons has a minimum of 569.25 tons to a maximum of 3757.05 tons. At any point of time, the heat load capacity must be larger than the cooling load performed by the chillers to prevent tripping.

Constraint 2: The temperatures of condenser water (supply and return) and wet bulb fall within the range of minimum and maximum values in Fahrenheit. Presently, the designated temperature for web bulb at inlet, T_{wb} is 82.4^{0}F, condenser water supply, T_{cws} is 88.7^{0}F and condenser water return, T_{cwr} is 98.6^{0}F.

Constraint 3: The cooling towers' mass flow of condenser water and air are modeled in m^3/h as m_w and m_a. They fall within the minimum and maximum designated capacity for optimized set points. Maximum air flow is achieved when all fan mode is 'on' that is 538446 m^3/h and water flow achieves its maxima when all pump mode is 'on' that is 5832 m^3/h.

After the initial population of 2160 chromosomes consists of the variables are evaluated for fitness, new population is generated using three genetic operators that are reproduction, crossover, and mutation. Each gene mutates with a probability of 0.1 and the crossover rate is 0.7. Once the operation setting of the single speed cooling tower fans and condenser water pumps have been optimized through the genetic operations, they are compared with the existing operating setting before been put in force. This is a safety measure to prevent the uncertainties of the genetic algorithm due to insufficient evolution time. If such a condition occurs, the system will operate at present set points without any changes until the next sampling period. In our project, the termination of genetic operations occur when there is NO better fitness value

Table 1. GA parameters setting

Initial Population	2160	**Crossover**	0.7
Mutation	0.1	**No of generation**	200

(the most minimum power consumption achieved) obtained from a generation. From our testing, we found out that a 200 generation would be when the fitness achieves its maxima.

6 Experimental Results

The simulation-optimization application focused on optimizing the control of the single speed cooling towers' fans and condenser water pumps on real time basis while the VSD controlled chillers are not interfered. The control affects the mass flow of air and water at both inlet and outlet of cooling towers which change the heat transfer effectiveness of cooling towers. The experiment aims to observe the optimized setting of cooling tower fans and pumps throughout 24 hours of a day based on ambient temperature, chillers' cooling load and cooling tower's heat transfer effectiveness. The average measured cooling tower efficiency based on Braun's model that takes into account the average ambient temperature at different hours of the day is shown in Fig. 3. All the data measured were displayed in a daily average because the tropical climate here does not have distinctive seasonal difference. The averaged data is representative for the whole year. However, the experiment has been simulated for daily data of three months. The ambient temperatures vary between 68.9^{0}F to 82.4^{0}F throughout 24 hours of a day. Due to the ambient temperature achieves its maxima from 13:00 to 17:00; the heat transfer effectiveness of the cooling towers is the highest at these hours. On the other hand, due to the efficiency of cooling tower vary throughout 24 hours; the chillers' cooling load also varies accordingly where it achieves its maximum at 17:00. The ambient temperatures that change throughout the day have affected the chiller cooling load and cooling tower efficiency. This is due to higher ambient temperature at cooling tower inlet during day time has increased the condenser water supply temperature as well and vice versa during night time. We measure the wet bulb temperature at cooling tower inlet hourly throughout the day, and then measure the mass flow of air, mass flow of water, condenser water supply and return temperatures in the cooling towers. We optimized the mass flow and temperatures based on the fitness value of the population through the operating state of the fans and pumps. The mass flows are optimized with the constraint that cooling load of chillers can be met by heat transfer capacity of the cooling towers at any hour of the day. When the ambient temperature is lower, the calculated efficiency of the cooling tower is lower assuming the condition that condenser water supply and return temperatures remain constant. However, the condenser water supply temperature also decreases when outdoor temperature decreases at any time of the day. With these temperatures drop, we optimize the mass flow of air and water of the cooling towers. With the optimization by evolutionary computation through genetic algorithm, we manage to optimize the operation state of six pumps and six fans within the condenser water loop. Fig. 3 also shows the hourly optimized power consumption of cooling towers (1st bar), chillers (2nd bar), optimized total power consumption (3rd bar), total power consumption before optimization (4th bar). The daily power consumption of these pumps and fans when fully operated is 17164.80kW daily which costs USD1716.48 (approx. USD 0.10 per kW presently) daily and USD626515.20 annually. With the optimization of these components, we are able to cut down the energy consumption by 5,960 kW daily and 2,175,400 kW annually. This will give a cost saving of USD217540 annually. This gives a 34.7% saving on the total cooling tower power consumption or 7% of total condenser water loop power consumption as compare to fixed operation approach used previously.

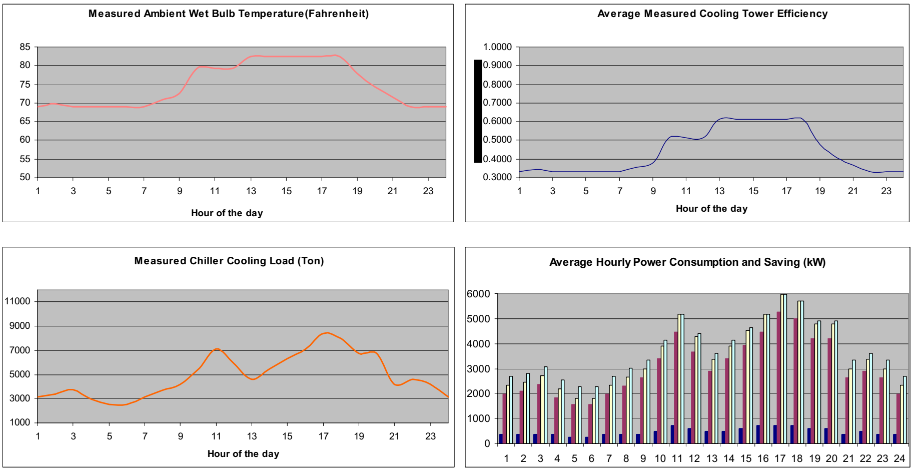


Fig. 3. Hourly Ambient Temperature, Chiller Cooling Load, Cooling Tower Efficiency and Power Consumption

7 Conclusions

This paper discussed a cost saving strategy through cooling towers in the condenser water loop. This strategy focuses on single speed control components in the loop where chillers and chilled water pumps which have variable speed drives are not modeled. The optimization cooling towers' water and air flows with fans and condenser water pumps through computational intelligence managed to cut down the total electricity consumed by cooling towers by 34.7% that is equivalent to USD 217,540 annually. The field data collection needs to be carried out for a longer term to observe any unusual behaviors of the parameters within the condenser water loop. With the more field data, optimization can be enhanced.

References

[1] Alcala, R., Casillas, J., Cordon, O., Gonzalez, A., Herrera, F.: A genetic rule weighting and selection process for fuzzy control of heating, ventilating and air conditioning systems. Engineering Applications of Artificial Intelligence 18, 279–296 (2005)

[2] Braun, J.E.: Methodologies for the design and control of chilled water systems. Ph.D. Thesis. University of Wisconsin (1988)

[3] Braun, J.E., Klein, S.A., Beckman, W.A., Mitchell, J.W.: Application of optimal control to chilled water systems without storage. ASHRAE Transactions 95(1), 663–675 (1989)

[4] Braun, J.E., Klein, S.A., Beckman, W.A., Mitchell, J.W.: Methodologies for optimal control of chilled water systems without storage. ASHRAE Transactions 95(1), 652–662 (1989)

[5] Chang, Y.C.: A novel energy conservation method-optimal chiller loading. Electric Power Systems Research 69(3), 221–226 (2004)

[6] Chow, T.T., Zhang, G.Q., Lin, Z., Song, C.L.: Global optimization of absorption chiller system by genetic algorithm and neural network. Energy and Buildings 34(1), 103–109 (2000)

[7] Fong, K.F., Hanby, V.I., Chow, T.T.: HVAC system optimization for energy management by evolutionary programming. Energy and Buildings 38, 220–231 (2006)

[8] Johnson, G.A.: Optimization techniques for a centrifugal chiller plant using a programmable controller. ASHRAE Transactions 91(2), 835–847 (1985)

[9] Lau, A.S., Beckman, W.A., Mitchell, J.W.: Development of computer control routines for a large chilled water plant. ASHRAE Transactions 91(1), 780–791 (1985)

[10] Lu, L., Cai, W., Chai, Y.S., Xie, L.: Global optimization for overall HVAC systems. Part I. Problem formulation and analysis. Energy Conversion and Management 46(7), 999–1014 (2005)

[11] MQuiston, F.C., Parker, J.D., Spitler, J.D.: Heating, ventilating and air conditioning: Analysis and design, 6th edn. John Wiley, Chichester (2005)

[12] Nabil, N., Samir, M., Mohammed, Z.: Self-tuning dynamic models of HVAC system components. In: Energy and Buildings, vol. 40, pp. 1709–1720. Elsevier, Amsterdam (2008)

[13] Sud: Control strategies for minimum energy usage. ASHRAE Transactions 90 (2) 247–277 (1984)

[14] Wang, S., Xu, X.: Effects of alternative control strategies of water-evaporative cooling systems on energy efficiency and plume control: A case study. Building and Environment 43, 1973–1989 (2008)

[15] Yao, Y., Lian, Z., Hou, Z., Zhou, X.: Optimal operation of a large cooling system based on an empirical model. Applied Thermal Engineering 24(16), 2303–2321 (2004)

[16] Yu, F.W., Chan, K.T.: Optimization of water-cooled chiller system with load-based speed control. In: Applied Energy, vol. 85, pp. 931–950. Elsevier, Amsterdam (2008)

Investigating Ensemble Weight and the Certainty Distributions for Indicating Structural Diversity

Lesedi Melton Masisi[1], Fulufhelo Nelwamondo[2], and Tshilidzi Marwala[1]

[1] University of the Witwatersrand, School of electrical and information engineering,
Private bag 3, Wits 2050, Johannesburg, South Africa
lesedi.masisi@students.wits.ac.za, tshilidzi.marwala@wits.ac.za
[2] Graduate School of Arts Sciences, Harvard University
Cambridge, Massachusetts, USA
nelwamon@fas.harvard.edu

Abstract. In this paper an investigation of the distribution of the weights and the biases of the Multilayered Perceptron is conducted, in particular the variance of the weight vector (weights and biases) with the aim of indicating the existence of the structural diversity within the ensemble. This will indicate how well the weight vector samples are distributed from the mean and this will be used to serve as an indicator of the structural diversity of the classifiers within the ensemble. This is inspired by the fact that many measures of ensemble diversity are focused on the outcomes and not the classifier's structure and hence may lose out in diversity measures that correlate well with ensemble performance. Three ensembles were compared, one non-diverse and the other two ensembles made diverse. The generalization across all the ensembles was approximately the same (74 % accuracy). This could be attributed to the data used. Certainty measures were also conducted and indicated that the non-diverse ensemble was biased, even though the performance across the ensembles was the same.

1 Introduction

The method of using more than one classifier has become an attractive method for improving the classification generalization. This method has resulted in many names such as, a committee of learners, mixture of experts, multiple classifier systems, classifier ensembles, etc. This has also resulted in immense research for proper aggregation schemes. A committee of classifiers has been found to be more efficient as opposed to using one classifier [1], [2], [3]. The advantages in ensemble systems is that the combination of classifiers improves upon the generalization of a single classifier [4]. However this is believed to be done by having individual classifiers that make different errors so that when they are combined they produce a much lower prediction error. This means that an ensemble that is composed of classifiers which have different decision boundaries [4]. Such an ensemble is considered to be diverse, hence the concept of ensemble

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 517–524, 2009.

diversity. This concept has lead to research into measures of ensemble diversity and has been noted as one of the key issues in improving ensemble performance [5], [6].

One of the most popular view of ensemble diversity is the error diversity. Whereby ensemble diversity is measured from the outcomes of the individual classifiers and has produced excellent work. Such work include, empirical measures of error diversity and statistical measures, the Q-Static, Correlation coefficient, the disagreement measure, double fault measure [7], the ambiguity decomposition [8] among many. However some of these measures have produced uncorrelated results when the performance of the ensemble is taken into account [9]. One of the conclusion one could introduce might be that ensemble diversity measures are meaningless, however this would be ignoring the fact that ensemble diversity does not have a formal definition among other factors, even though its been observed that many researchers focus on the outcomes for measures of ensemble diversity. Immense research has been made to correlate the ensemble diversity measures for building efficient ensembles [10], [9] among others and this research is still an open topic.

This paper aims to introduce a new way of looking at ensemble diversity. This is because the current view on ensemble diversity has not still produced a robust generic measure of ensemble diversity that relates well with performance of the ensemble. Measures of ensemble diversity still remain to be an attractive research area [9]. The ensemble diversity measure of concern, focuses at the architectural parameters of the artificial neural network to indicate ensemble diversity. Firstly this is inspired by the fact that current measures of ensemble diversity when related to ensemble generalization have produced inconclusive results when trying to use the diversity measures to build efficient ensembles [9]. Secondly as mentioned ensemble diversity has no formal definition. Hence Ensemble Structural Diversity (ESD) is proposed witch is hoped at broadening and bringing more knowledge to the scope of ensemble diversity measures. ESD is when the ensemble is composed of different structural parameters of the classifier within the ensemble. The ensemble could be composed of classifiers with different hidden nodes, learning rates, different mapping functions, etc. There has been other studies on structural diversity and ensemble generalization, whereby ensemble structural diversity did improve ensemble generalization. The results also showed that high structural diversity could create poor generalization [11].

This study analysis the distributions of the weight vectors (weights and biases) of the MLP's composed within the ensemble. In particular the distribution parameter of concern is the variance which also leads to knowledge of the standard deviation of the weight vector samples from the mean. This was inspired by the fact that the number of hidden nodes controls the complexity of the Neural Network (NN), which translates to the structure of the Multilayered Perceptron (MLP). Aimed objectives of viewing ensemble diversity in terms of the structure and not the outcomes is to: Broaden the research scope for ensemble diversity measures, add new understanding to ensemble diversity and possibly lead to other measures not focused on the classifier's outcomes to measure ensemble diversity, lead to a unique definition of ensemble diversity hence robust measures for ensemble diversity. The rest of the paper attempts to meat these objectives. A method of voting was used to fuse or aggregate the individual classifiers for a final decision of the ensemble. The interstate conflict data was used for demonstrating the concept of structural diversity in this paper. The sections of this paper are organized as follows, the following section deals with, ensemble diversity induction, Structural diversity, methodology, results and discussion and then the conclusion.

2 Ensemble Diversity

Two methods for inducing ensemble diversity of the classifiers are conducted. The first method uses the Bagging algorithm, short for boostrap aggregation, and the second method is by using classifiers with different number of hidden nodes, this method is one of the basis of the concept of ESD. Bagging is considered to be one of the earliest ensemble algorithm and one of the simplest to implement [12]. The bagging algorithm trains the ensemble of classifiers by exposing the individual classifiers to a randomly chosen sample from the training data. These classifiers are normally known as hypotheses or base learners, usually referred to as weak learners since they do not learn the whole data set but just a sample of it [4]. In this way the classifiers learn different domains of the problem and hence diversity can be induced.

Structural diversity would also imply that the classifiers have variations in the distributions of the classifier weights. The weight vector samples of the MLP are initialized from a Gaussian distribution and hence it will be expected that initially the weight vector samples have the same variance and the mean. This would mean that it is the training scheme of the ensemble that greatly influences the distribution parameters of the weight vector samples of the MLP. Hence using different complexities will also affect the weight vector samples differently and hence a study on the distribution of the weight vector might shed light to how the parameters of the distributions relate to ensemble generalization.

However within the ensemble diversity studies, it is not only the measures of diversity that are of concern but also the aggregation methods. Kuncheva [5], among many, has looked at the relationship between combination methods and measures of diversity and has found that certain measures of diversity correlated with certain aggregation schemes. It was also noted that the correlation observed had strong dependency on the data used [5]. This shows the complexity of the ensemble diversity studies which has also been noted by [8], [5], [13], [14], [3], et al. This complexity is also expected for the ESD measures. This implies that developing a good measure and having a good aggregation scheme does not normally go hand in hand. However for the sake of prove of concept only the majority vote scheme is considered due to its wide use.

3 Methodology

From error diversity measures, it was found that diversity reduces the variance in the decomposition of the error measure [8] and improves ensemble generalization. Two methods are used to induce diversity, the first one is by the use a training algorithm (bagging) and the second one is by having a committee of classifiers with different number of hidden nodes. Then a measure of diversity that looks at the variances of the classifier weights vector (weights and biases) is conducted. This measure is then compared to the generalization performance of the ensembles. Figure 1 shows the steps taken for the process mentioned. Only an ensemble containing five classifiers was used. Five classifiers were only considered so that computational cost was minimized and the number was made odd so that there were no ties during voting, for the final decision. The classifiers initially had 8 hidden nodes with a linear activation function. However when inducing diversity via the architecture of the classifiers, the hidden nodes where then varied randomly between 8 and 21. This was so that they were not biased.

Certainty or confidence measures on the individual outcomes of the classifiers were done so that more knowledge could be gained on the generalization of the ensembles. The variance of certainties from the five classifiers will be used to give a more precise indication of the diversity of the ensemble. This means if there is no variance then the ensemble is highly certain which means that it could be highly biased. The certainties are measured from the outputs of the individual classifiers for the outputs of the classifiers are taken as a probability measure. Normally in a binary classification problem a 0.5 output is treated as a 1. However, in this paper a 0.5 output was not immediately rounded off to a 1, since its confidence measure is a zero, as can be seen from (1). The 0.5 output is taken as being the same as tossing a coin and thus its final outcome would either be a 0 or a 1. The certainty is symmetrical about the 0.5 output from the classifier. To illustrate equation (1), a 0.9 output from a classifier will have a confidence of 0.8 and a final classification output of 1. An output of 0.4 will be assigned a confidence of 0.2 and would mean a 0 for the final outcome (after rounding off). This equation is inspired from the Dynamically Averaging Networks (DAN) by Daniel Jimenez [15] and it was modified in this paper.

$$C(f(x_i)) = |2f(x_i) - 1| \tag{1}$$

Where, $f(x_i)$ is the immediate output from a classifier with input x_i before being processed into binary and $C(f(x_i))$ is the certainty of the processed input data to a certain class.

$$S(x) = \frac{\sum_{f(x_i)=y_i} C(f_i)}{\sum C(f)} \tag{2}$$

Where, S is the sum of all the classifiers which won the vote for an input data x_i, y_i is the correct output value from the data and the $\sum C(f)$ is the normalizing factor. The last step is to calculate the variance of the S vector for all the data samples that were correctly classified. Hence the variance of the distribution of

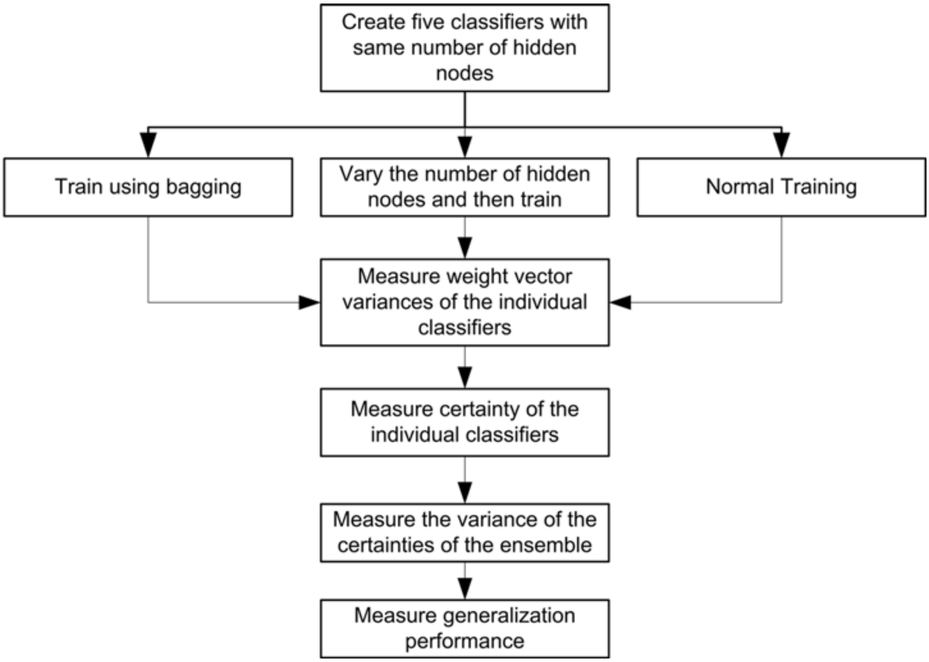

Create five classifiers with same number of hidden nodes
Train using bagging
Vary the number of hidden nodes and then train
Normal Training
Measure weight vector variances of the individual classifiers
Measure certainty of the individual classifiers
Measure the variance of the certainties of the ensemble
Measure generalization performance

Fig. 1. Flow diagram on diversity analysis

the certainties of the correctly classified data samples. See equation(2) for the calculation of the certainty of the classifiers that won the vote. This certainty is the confidence of the overall classifiers for that particular data sample. The certainties were normalized between 0 and 1, representing high to low certainties respectively. The tests were done on the interstate conflict data [16].

The data has 7 features and a binary output. A 0 represented a conflict and a 1 peace. The data was conditioned such that it had approximately 50/50 conflict and peace cases. The training data was composed of 1006 and 869 for training and testing respectively. The data was normalized between 0 and 1 so as to have same weighting for all the input features. An ensemble would be considered diverse if it had different variances on the distributions of the weights vector (weights and biases) between classifiers. A Multi Layered Perceptron (MLP) was used for all the experimentation. The certainty measures were conducted on the test data set where else the weights vector variance measures, were conducted after the classifiers were trained.

4 Results and Discussion

Two methods of inducing diversity have been studied, the Bagging and parameter change of the ensemble of classifiers. The structural diversity was observed over the two diverse ensembles. The results on the accuracies do not show any much of a difference between the different ensembles. They all produced accuracies of approximately 74 %. However it is evident from the variances of the weights vector that the ensemble is diverse. The variances on the weight vector due to the bagging algorithm were still on a close range, see table 1. This showed that less diversity was induced on this ensemble.

However the variances due to structural diversity (changing of hidden nodes) produced classifiers that had the vector weight variaces significantly different within the committee, see table 1 on the second column. Intutively one would conclude that this ensemble was more diverse as compared to the ensemble trained via bagging. But the bagged ensemble produced better generalization performance, see 2, as compared to the other ensemble (diverse due to structure). This might mean that the observing the vector weights as a measure of ensemble diversity might not relate well with the generalization performance of the ensemble.

The significant weight vector variances on the ensemble (among the classifiers) with different number of hidden nodes, can be attributed to the use of different number of hidden nodes. This then shows that the use of the weight vector distributions might not be a a good method to correlate diversity and generalization. This is one of the biggest disadvantages of measuring diversity from the structural point of view. For one can develop a good measure but then loose out on using the measure to predict the generalization performance of the ensemble, as noted.

According to these results, see table 1 and 2, when the ensemble is non-diverse then the variance of the certainties is zero. That means the non-diverse ensemble is extremely certain that there is no variation. Intuitively this made sense for an ensemble that is non-diverse due to the ensemble being biased. This means that this certainty measure should not be confused with the confidence of reducing risk in classification.

Table 1. Variances of the diverse and non diverse ensembles

Bagged(σ^2)	Nodes(σ^2)	Non-Diverse(σ^2)
0.25915	0.4582	0.26391
0.30675	0.23119	0.26391
0.27167	0.22072	0.26391
0.23999	0.18754	0.26391
0.29347	0.16668	0.26391

The confidence in this context measured the extent to which the individual classifiers believed to have been correct not necessary that the classification was correct. This shows how structural diversity measures can better bring more understanding to the classification problems. These results show that the data used was not complex enough and one classifier would be adequate for this problem. This is concurrent with literature that diversity can both be harmful or beneficial [17]. Further work can be done by using different aggregation schemes even the Dynamic Average Networks (DAN) for understanding the structural variation of the classifiers. Structural diversity measures could also be attempted in other artificial machines.

Table 2. Acuracies and variance measures

Acc (Bagged)	74.914
Acc (Nodes)	74.569
Acc(nondiverse)	74.338
$\sigma^2(C_{div}(f(x)))$	0.0064141
$\sigma^2(C_{non-div}(f(x)))$	0

5 Conclusion

This paper presented concepts inspired from statistical methods and certainty to better understand the structural diversity of the ensemble. A different assessment of ensemble diversity has been presented as opposed to looking at the classifier outcomes to measure ensemble diversity. The weight vector of the classifiers have been assessed for indicating ensemble diversity. Due to looking at the outcomes and not at the structure misleading judgments about the ensemble diversity might be taken and hence poor correlation judgments on the generalization of the ensemble and ensemble diversity. This was observed from having a structurally diverse ensemble having almost the same generalization as the non-diverse ensemble. However this could be attributed to the size of the data and the data used. Certainty or confidence variance of zero was recorded due to a non-diverse ensemble which indicated that the non-diverse ensemble was biased. Knowledge on viewing ensemble diversity form structural point of view adds more knowledge on distributions of the vector weights of the diverse ensembles and hence on the ensemble diversity research community. Measuring ensemble diversity from the vector of weights might be meaningful but correlating this measure with ensemble generalization might not be meaningful. A formal definition of ensemble diversity still remains an open discussion. More work can still be done in changing the number of the classifiers within a committee and by using other data sets. Classifiers have a number of parameters depending on the artificial machine used, which leaves the search for other methods of measuring structural diversity for exploration.

References

1. Igelnik, B., Pao, Y., Leclair, S., Shen, C.: The ensemble approach to neural-network learning and generalization. IEEE transactions on neural networks 10(1), 19–30 (1999)
2. Islam, M.M., Yao, X., Nirjon, S.M.S., Islam, M.A., Murase, K.: Bagging and boosting negatively correlated neural networks. IEEE transactions on systems, man, and cybernetics. Part B 38, 771–784 (2008)
3. Kuncheva, L.I., Skurichina, M.: An experimental study on diversity for bagging and boosting with linear classifiers. Information Fusion 3, 245–258 (2002)
4. Polikar, R.: Ensemble based systems in decision making. IEEE Circuits and Systems Magazine 6(3), 21–45 (2006)
5. Shipp, C.A., Kuncheva, L.I.: Relationships between combination methods and measures of diversity in combining classifiers. Information Fusion 3(2), 135–148 (2002)

6. Lam, L.: Classifier Combinations: Implementations and Theoretical Issues. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 78–86. Springer, Heidelberg (2000)
7. Gal-Or, M., May, J.H., Spangler, W.E.: Assessing the predictive accuracy of diversity measures with domain-dependent asymmetric misclassification costs. Information Fusion 6(1), 37–48 (2005)
8. Brown, G.: Diversity in neural network ensembles. Ph.D. thesis, School of Computer Science, University of Birmingham (2004)
9. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles. Machine Learning 51, 181–207 (2003)
10. Tumer, K., Ghosh, J.: Error correlation and reduction in ensemble classifiers. Connection Science 8, 385–404 (1996)
11. Masisi, L., Nelwamondo, F.V., Marwala, T.: The effect of structural diversity of an ensemble of classifiers on classification accuracy. In: IASTED International Conference on Modelling and Simulation, Africa-MS (2008)
12. Breiman, L.: Bagging predictors. Machine Learning 24, 123–140 (1992)
13. Cantu-Paz, E., Kamath, C.: An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. Systems, Man, and Cybernetics-Part B: Cybernetics 35, 915–927 (2005)
14. Izrailev, S., Agrafiotis, D.K.: A method for quantifying and visualizing the diversity of qsar models. Journal of Molecular Graphics and Modelling 22, 275–284 (2004)
15. Jimenez, D.A., Walsh, N.: Dynamically weighted ensemble neural networks for classification. In: Proceedings of the 1998 International Joint Conference on Neural Networks (1998)
16. Marwala, T., Lagazio, M.: Modeling and controlling interstate conflict. In: IEEE International Joint Conference on Neural Networks, Budapest, Hungary, pp. 1233–1238 (2004)
17. Kuncheva, L.I., Whitaker, C.J., Shipp, C.A., Duin, R.P.W.: Is independence good for combining classiffiers? In: Proceedings of 15th International Conference on Parttern Recognition, vol. 2, pp. 168–171 (2000)

Part V

Neural Networks and Pattern Recognition

Dynamic Programming Stereo on Real-World Sequences

Zhifeng Liu and Reinhard Klette

The *.enpeda..* Project, The University of Auckland
Auckland, New Zealand

Abstract. This paper proposes a way to approximate ground truth for real-world stereo sequences, and applies this for evaluating the performance of different variants of dynamic programming stereo analysis. This illustrates a way of performance evaluation, also allowing to derive sequence analysis diagrams. Obtained results differ from those obtained for the discussed algorithms on smaller, or engineered test data. This also shows the value of real-world testing.

1 Introduction

Vision-based driver assistance is one of the largest challenges in current applied computer vision. Algorithms have to process real-world stereo sequences (e.g., under all possible weather conditions) in real time. Car crash tests are performed based on very strict international standards; the same is expected soon for tests of vision-based driver assistance modules. This paper deals with real-world stereo sequences.

There are not yet many reference sequences available for comparative performance evaluation. We refer in this paper to Set 1 (provided by Daimler AG) of the *.enpeda..* sequences,[1] as described in [4]. These seven stereo sequences are taken with two Bosch (12-bit, gray-value) night vision cameras. Each sequence contains 250 or 300 frames (640×481), and features different driving environments, including highway, urban road and rural area. Camera calibration is used for geometric rectification, such that image pairs are characterized by standard epipolar geometry as specified in [3].

Intrinsic camera parameters and extrinsic calibration parameters for left and right camera (also in relation to the car) are provided. The vehicle's movement status is also given for each frame. We discuss a way to approximate partial ground truth from these sequences.

2 Methodology

To evaluate the performance of a stereo algorithm, and understand how its parameters affect results, we need a quantitative way to measure the quality of calculated stereo correspondences or motion vectors.

[1] http://www.mi.auckland.ac.nz/EISATS

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 527–534, 2009.

Approximated ground truth. We assume a planar road surface for a selected sequence of stereo frames. These can be short sequences of just (say) 20 stereo frames. Here we illustrate for sequences of the given length of 220 to 300 frames. – We consider the test sequences to be ego-motion compensated [2], which means that the horizon is always parallel with the row direction in the images. We conclude that pixels on the same image row have the same depth value if a projection of the planar road surface.

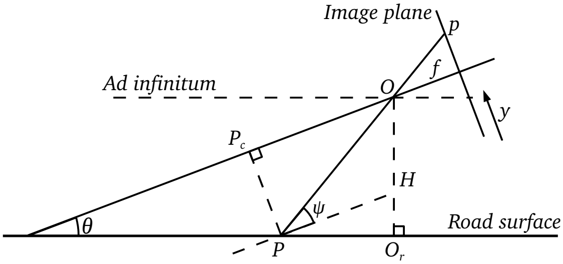


Fig. 1. Projection of a point P of the road surface

A side-view of the camera setting is shown in Figure 1, where θ is the known tilt angle, P is a road surface point which is projected into $p = (x_p, y_p)$ on the image plane, H is the height of the camera. It follows that

$$Z = d_e(OP_c) = d_e(OP) \cos \psi = \frac{H}{\sin(\theta + \psi)} \cos \psi \tag{1}$$

According to standard stereo projection equations [3], the disparity d can be written as

$$d = \frac{b \cdot f}{Z} = \frac{b \cdot f}{\frac{H}{\sin(\theta+\psi)} \cos \psi} \tag{2}$$

where angle ψ can be calculated as follows, using focal length f and pixel coordinate y_p in the image:

$$\psi = \arctan \left(\frac{(y_p - y_0) s_y}{f} \right) \tag{3}$$

Here, y_0 is the y-coordinate of the principal point, and s_y is the pixel size in y-direction. We can also compute the y-coordinate of a line that projects to infinity

$$y_{inf} = \frac{y_0 - f \cdot \tan \theta}{s_y}$$

This is the upper limit of the road surface, and points on it should have zero disparity (if no objects block the view).

Figure 2 illustrates the process of generating an approximated disparity map on road surface areas, also using manual input for a conservative outline of the road area in a given image. In the given camera setting (of the seven sequences), there is a yaw angle (0.01 radian) which makes the cameras looking a little bit

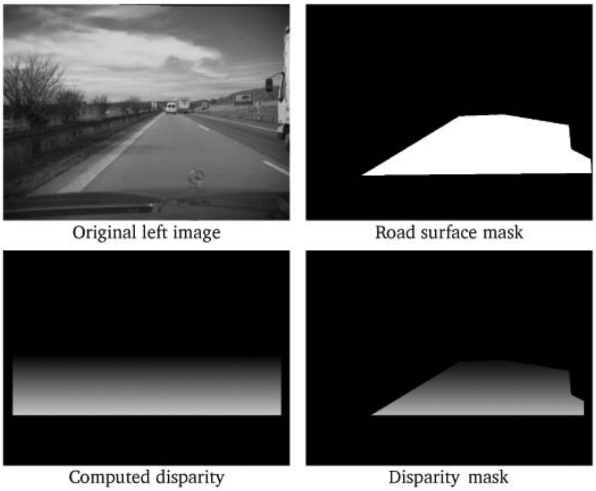


Fig. 2. Generation of a disparity mask: input image, manually generated mask, depth map of a planar road, and resulting disparity mask

to the left. This angle can be ignored because it only defines the right camera to be about 3 mm behind the left camera.

See Figure 2 and assume a given pair of corresponding points, with disparity d. By Equation (2) we have that the tilt angle can be written as follows:

$$\theta = \arcsin\big(\frac{H\cos\psi \cdot d}{b \cdot f}\big) - \psi \qquad (4)$$

where ψ is as given in Equation (3). Table 1 shows the estimated tilts for the seven sequences.

Error metrics. The general approach of stereo evaluation is to compute error statistics based on given ground truth. (Note that any ground truth comes with some measurement error; ground truth is not truth.) We use the same error measurements as on the Middlebury stereo website [6], namely the *root mean*

Table 1. Results of tilt angle estimation for the given seven sequences

Sequence name	Tilt angle (radian)
1: Construction-Site	0.016
2: Save-Turn	0.013
3: Squirrel	0.021
4: Dancing-Light	0.061
5: Intern-on-Bike	0.062
6: Traffic-Light	0.069
7: Crazy-Turn	0.060

squared error between the disparity map $d(x, y)$ and the ground truth map $d_T(x, y)$, defined as follows:

$$E_R = (\frac{1}{n} \sum |d(x, y) - d_T(x, y)|^2)^{\frac{1}{2}} \tag{5}$$

where n is the total number of pixels, and the percentage of *bad matching pixels*, defined as follows:

$$E_B = \frac{1}{n} \sum(|d(x, y) - d_T(x, y)| > \delta_d) \tag{6}$$

where δ_d is the threshold of disparity tolerance.

Tested approaches. We evaluate dynamic programming stereo, using variations of sources as available on [5]. We run a standard stereo dynamic programming (DP) approach (e.g., see [3]) on the given seven sequences; see Table 2 for evaluation results. Sequence 1 returns smallest RMS errors and bad matching percentages. In contrast, Sequence 6 returns the largest error values out of the seven sequences.

DP is then also modified by using some spatial propagation of disparities (from previous row to the current row, with a weight of 20%) or some temporal propagation of disparities (from the same row in the previous pair of frames, again with a weight of 20%). Furthermore, we run Birchfield-Tomasi (BT, designed to be an improvement of standard stereo DP).

3 Results and Discussion

The experiment on Sequence 7 is only performed on the first 220 frames, instead of the total number of 250, because the road surface is reduced to a very small area after the ego-vehicle makes a large turn to the left.

The DP algorithm with spatial propagation (DPs) takes 20% of the disparity value from the previous scanline into the final result. In other words, we apply

$$d'_{y,t} = (1 - \lambda_1) d_{y,t} + \lambda_1 d_{y-1,t} \quad \text{where} \quad \lambda_1 = 0.2$$

Table 2. Mean RMS error values (5) and mean bad matching percentages (6) for the standard DP algorithm

Sequence name	Number of frames	RMS	Bad matches
1: Construction-Site	300	0.020	2.7%
2: Save-Turn	300	0.023	8.5%
3: Squirrel	300	0.023	23.1%
4: Dancing-Light	250	0.068	21.4%
5: Intern-on-Bike	250	0.064	17.5%
6: Traffic-Light	250	0.072	44.8%
7: Crazy-Turn	220	0.056	35.8%

Table 3. Mean RMS error values (5) and mean bad matching percentages (6) for DP with temporal propagation

Sequence name	Number of frames	RMS	Bad matches
1: Construction-Site	300	0.020	1.9%
2: Save-Turn	300	0.018	3.3%
3: Squirrel	300	0.022	17.8%
4: Dancing-Light	250	0.068	19.2%
5: Intern-on-Bike	250	0.064	16.4%
6: Traffic-Light	250	0.072	45.3%
7: Crazy-Turn	220	0.054	32.9%

Table 3 illustrates the DP algorithm with temporal propagation (DPt), which uses

$$d'_{y,t} = (1 - \lambda_2) d_{y,t} + \lambda_2 d_{y,t-1} \quad \text{where} \quad \lambda_2 = 0.2$$

DP with temporal and spatial propagation (DPts) uses

$$d'_{y,t} = (1 - \lambda_1 - \lambda_2) d_{y,t} + \lambda_1 d_{y-1,t} + \lambda_2 d_{y,t-1}$$

where $\lambda_1 = 0.1$ and $\lambda_2 = 0.1$.

Figure 3 shows a comparison between DP and its variants, for all the frames of Sequence 1. Result show that spatial propagation causes more errors than the standard DP algorithm. Of course, the road surface is represented as a slanted plane whose disparity map changes smoothly from 0 (at infinity) to about 50. This particular geometry violates the assumption of spatial propagation. (Spatial propagation might be still of interest within object regions.)

Time propagation shows (for all seven sequences) an obvious improvement by keeping the RMS error about at the local minimum of the standard DP. Of

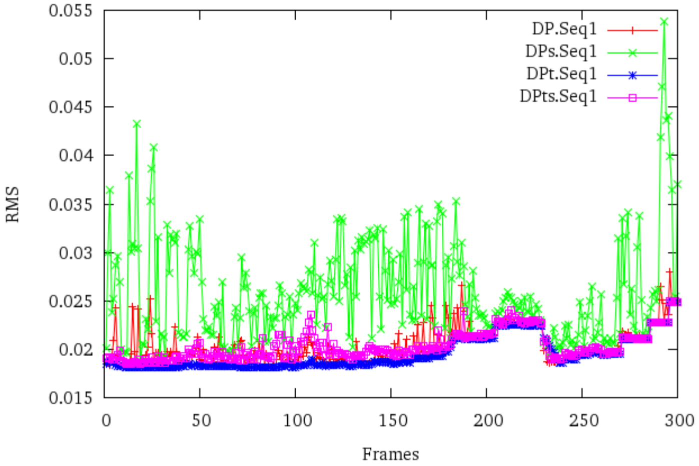


Fig. 3. Comparing RMS error (5) results between DP and its variants.

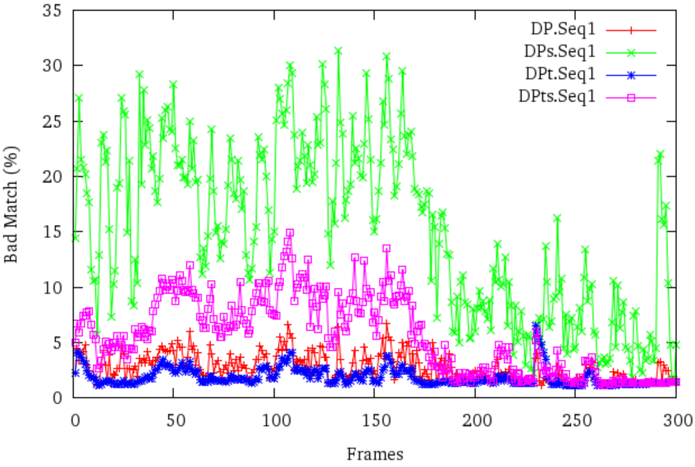


Fig. 4. Percentages of bad matches (6) for DP and its variants

course, driving on a plane means that disparity values should remain constant, and any deviation from this may be used to detect a change, such as a 'bumpy' road. DPts, the combined propagation method, shows a similar outcome as DP without any propagation.

A comparison with respect to the second quality metric (percentage of bad matches) is shown in Figure 4. Again, temporal propagation does have a positive effect, and spatial propagation is worsening results. (Note that this evaluation is only restricted to the road surface area.)

Now we discuss the Birchfield-Tomasi algorithm (BT). Table 4 shows evaluation results of the BT algorithm (with an occlusion penalty of 25 and a reward parameter of 5). Compared with DP techniques, the disparity maps and the quality metrics indicate bad results for BT; disparity values are typically incorrect on the road surface.

This bad performance may be due to the following two reasons. First, the BT algorithm is developed on the concept of the existence of depth discontinuities.

Table 4. Mean RMS error values (5) and mean bad matching percentages (6) for the BT algorithm

Sequence name	Number of frames	RMS	Bad matches
1: Construction-Site	300	0.09	61%
2: Save-Turn	300	0.11	97%
3: Squirrel	300	0.11	81%
4: Dancing-Light	250	0.13	99%
5: Intern-on-Bike	250	0.12	95%
6: Traffic-Light	250	0.14	100%
7: Crazy-Turn	220	0.11	99%

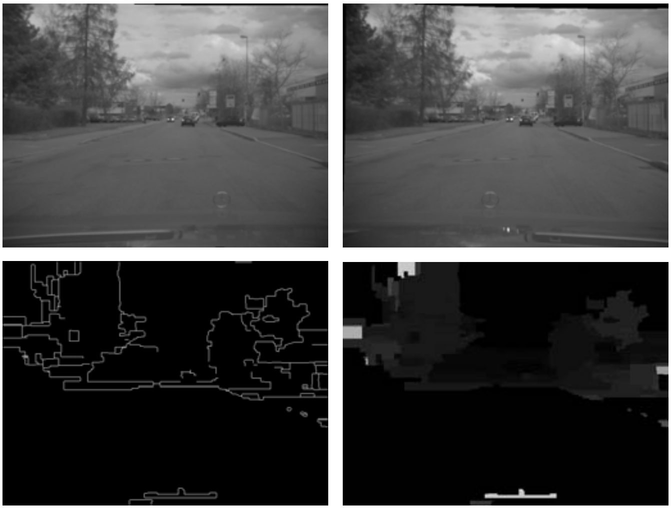


Fig. 5. The performance of the BT algorithm depends on depth discontinuities. Upper left: left image of a stereo input pair. Upper right: road mask. Lower left: depth discontinuity image. Lower right: calculated disparity map.

However, depth discontinuities may not exist in many real world situations, such as on the road. For example, Figure 5 shows that there is no edge detected close to the car.

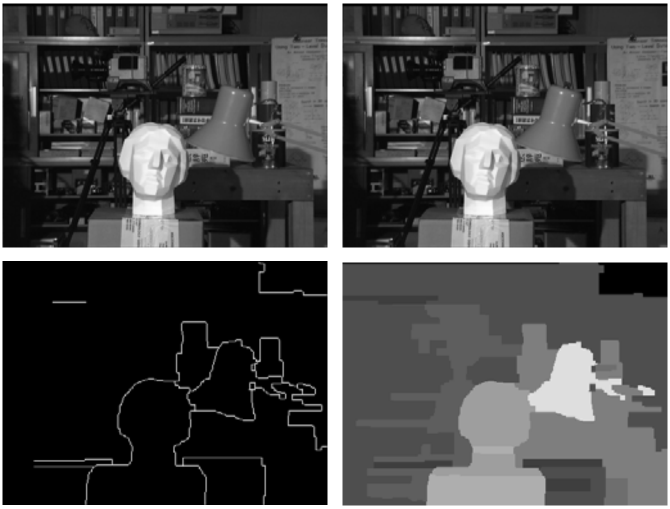


Fig. 6. Upper row: a stereo input pair of the Tsukuba sequence. Lower left: depth discontinuity image. Lower right: calculated disparity map using BT.

Second, the BT algorithm uses a disparity propagation method to fill in untextured areas, both in horizontal and vertical directions. However, within the road surface area, the true disparities only change very smoothly in vertical direction.

We also run the BT algorithm (as implemented) on Middlebury stereo data, see Figure 6 for the Tsukuba test sequences. The same problem, as widely visible in the road scenes, occurs in the untextured area in the upper right corner. Except of this minor image region, BT appears here to be of advantage in general.

4 Conclusions

The difficulty for the evaluation of stereo techniques on real-world sequences is the lack of ground truth. This problem is partially solved by approximating the 3D geometry of the road.

The paper illustrated the use of these on-road estimates for evaluating the performance of variants of dynamic programming stereo on real-world sequences.

Further approximate ground truth (such as estimated poses of simple objects, such as rectangular faces in the scene) might be accumulated, to go, step by step, towards a 3D modeling of the actually recorded real scene. Of course, some objects or features are not of interest with respect to applications such as driver assistance or traffic monitoring.

The order of the algorithms' performance is clearly inconsistent to that reported on the Middlebury stereo or optical flow website. This difference shows the necessity for establishing performance evaluation methods on (various) real-world sequences ('Computer Vision beyond Middlebury' - without neglecting the very positive influence these engineered test examples had and have; but it is certainly critical if overdoing one particular way of evaluation).

References

1. enpeda. Image Sequence Analysis Test Site, http://www.mi.auckland.ac.nz/EISATS
2. Franke, U., Gehrig, S., Badino, H., Rabe, C.: Towards optimal stereo analysis of image sequences. In: Sommer, G., Klette, R. (eds.) RobVis 2008. LNCS, vol. 4931, pp. 43–58. Springer, Heidelberg (2008)
3. Klette, R., Schlüns, K., Koschan, A.: Computer Vision. Springer, Singapore (1998)
4. Liu, Z., Klette, R.: Performance evaluation of stereo and motion analysis on rectified image sequences. Technical report, Computer Science Department, The University of Auckland (2007)
5. Open Source Computer Vision Library, http://www.intel.com/research/mrl/research/opencv/
6. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. Computer Vision 47, 7–42 (2002)

Happy-Sad Expression Recognition Using Emotion Geometry Feature and Support Vector Machine

Linlu Wang, Xiaodong Gu[*], Yuanyuan Wang, and Liming Zhang

Department of Electronic Engineering, Fudan University, Shanghai 200433, P.R. China
linlu_wang@yahoo.com.cn, guxiaodong@263.net, xdgu@fudan.edu.cn

Abstract. Currently human-computer interaction, especially emotional interaction, still lacks intuition. In health care, it is very important for the medical robot, who assumes the responsibility of taking care of patients, to understand the patient's feeling, such as happiness and sadness. We propose an approach to facial expression recognition for estimating patients' emotion. Two expressions (happiness and sadness) are classified in this paper. Our method uses a novel geometric feature parameter, which we call the Emotion Geometry Feature (EGF). The active shape model (ASM), which can be categorized mainly for non-rigid shapes, is used to locate Emotion Geometry Feature (EGF) points. Meanwhile, the Support Vector Machine (SVM) is used to do classification. Our method was tested on a Japanese Female Facial Expression (JAFFE) database. Experimental results, with the average recognition rate of 97.3%, show the efficiency of our method.

1 Introduction

During the past three decades, facial expression analysis has attracted more and more attention in the computer vision field for its many applications, such as human-machine interaction, image understanding, synthetic face animation [1], and web services. Facial expressions reflect not only emotions, but also other mental activities, social interaction and physiological signals [2]. Therefore, in health care, it is important for a medical robot that helps the patient, to recognize the patients' emotion through facial expression [3].

Even though much work has been done, facial expression recognition with a high accuracy is very difficult due to the non-rigidity and complexity of facial expression. The facial expressions were generally defined by psychologists as a set of six basic facial expressions [4], including anger, disgust, fear, happiness, sadness, and surprise. In this paper we mainly recognize the happy expression and sad expression.

Many research efforts have been put into facial expression recognition. A survey on the facial expression recognition can be found in [5] and [6]. In [6], Pantic and Rothkrantz surveyed the research work done in automating facial expression analysis. In [5], Fasel and Luettin introduced the most prominent automatic facial expression analysis methods and systems presented in the literature. In addition, they also discussed facial motion and deformation extraction approaches and classification

[*] Corresponding author.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 535–542, 2009.

methods in [5]. According to the approach used for facial-expression feature extraction, methods about facial expression recognition can be distinguished as the feature-based method [7, 8] and the template-based method [9, 10].

Since the Facial Action Coding System (FACS), which is a system designed to describe changes in the facial expression in terms of observable activations of facial muscles [12], was developed by Ekman and Friesen [11] to code facial expressions by Action Units (AUs), several research efforts [12, 13] have been made to recognize facial expression based on the Facial Action Coding System (FACS). The Active Shape Model proposed by Cootes [14], is one of the sophisticated deformable template models to detect facial features [15]. However traditional facial features extracted by ASM cannot accurately represent facial expressions.

So, motivated by FACS and ASM, we propose a model of facial expression recognition for estimating patients' emotion based on EGF, which we introduce in this paper. Emotion Geometry Feature (EGF) concentrates on expression features, rather than facial features, to form a facial recognition system.

The rest of the paper is organized as follows. Section 2 introduces our facial expression recognition model. In this model, we introduce a novel expression feature recognizer, EGF. The Active Shape Model, which is used to locate Emotion Geometry Feature (EGF) points, is also described simply in Section 2. Using the Japanese Female Facial Expression (JAFFE) database as the test data, experimental results and analysis are shown in Section 3. Conclusions are drawn in Section 4.

2 Our Facial Expression Recognition Approach

Fig. 1 shows the flow diagram of the proposed model. Our model can be divided into three parts: one for the creation of The Active shape Model with a set of training images; two for the location of facial expression features; three for the extraction of EGF and the classification of two expressions.

ASM created by the training samples can search new facial expression feature points in the testing images. After the location of feature points we make efforts to analyze the difference between happiness expressions and sadness expressions and extract EGF including shape coordinates, the distance between upper lip and lower lip and the geometric angle between the line $\overline{\mathbf{O}_i\mathbf{A}_i}$ and the line $\overline{\mathbf{O}_i\mathbf{B}_i}$ (Fig. 3). Before applying SVM, it's very important to scale the input parameters [16]. An important objective is to avoid numerical difficulties during the calculation [17]. Ultimately classified facial expressions are obtained.

2.1 Overview of ASM

Cootes et al. [14] introduced the Active Shape Model (ASM), a method of fitting a set of local feature detectors to an object. The ASM procedure starts with a prior knowledge about the object shape, so that the model can extract the object shape in a new image by locating the outline of the object. So, we can divide the ASM procedure into two steps made up of modeling and matching.

A linear shape model can be generated by a set of manually labeled training image points, the formula is,

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{\Phi}_s \mathbf{b}_s, \tag{1}$$

where $\mathbf{x}$ is the synthesized shape, $\bar{\mathbf{x}}$ is the mean of shapes, and $\mathbf{b}_s$ is a set of shape model parameters, and $\mathbf{\Phi}_s$ is the matrix which consists of eigenvectors of the covariance matrix of the training set, and is an orthogonal matrix.

Virtually, the matching is to locate the outline points of the object by searching in the testing image by using the model built. In searching, we can obtain the optimal parameters for location and shape of the face by comparing the reference model from the training set to a new test image [24].

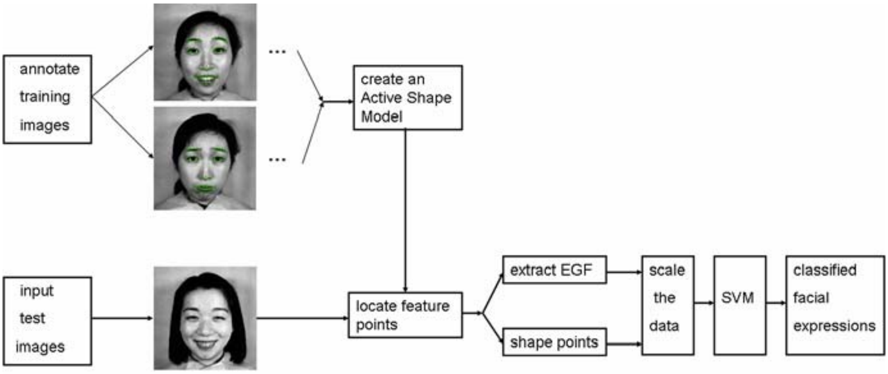


Fig. 1. The flow diagram of the proposed model

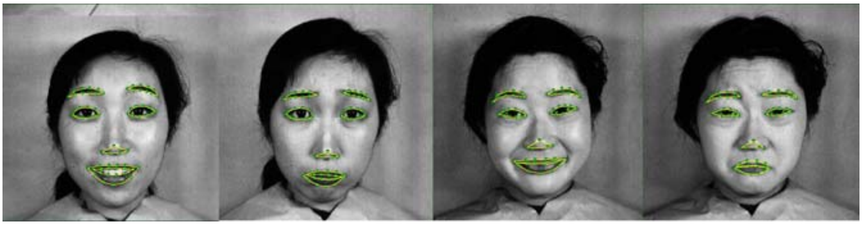


Fig. 2. The examples of labeled training data, and each of them is labeled with 66 points around the mouth, the nose tip, eyes, and eyebrows

2.2 Proposed Emotion Geometry Feature

In this section we compare happiness expressions with sadness expressions, and introduce the concept of the Emotion Geometry Feature (EGF) containing the angle and the distance, and the position of shape features. Here, the angle feature and the distance feature are the novel ones we mainly propose in terms of facial expression recognition in this paper, and the position information can be found in other papers.

1). a and b in Fig. 3 illustrate that a mouth which is deformable in a happiness image is wider than one in a sadness images, but the nose tip is rigid in different circumstances. So we consider the nose tip, and two corners of the mouth as three vertices of a triangle. The coordinates of three vertices are,

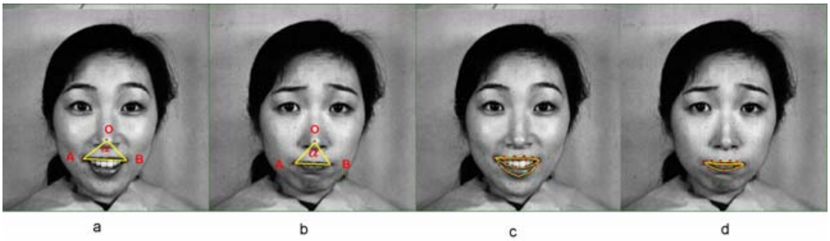


Fig. 3. Images a, b are labeled O, A, B on nose tip, and corners of mouth, respectively, and images c, d are labeled around the mouth. a and c are images with happiness expression, meanwhile, b and d are samples with sadness expression

$$\mathbf{O}_i = (\mathrm{x}_i^0, y_i^0)^T, \mathbf{A}_i = (\mathrm{x}_i^1, y_i^1)^T, \mathbf{B}_i = (\mathrm{x}_i^2, y_i^2)^T, i = 1, 2, \cdots, N, \tag{2}$$

where N is the number of facial images, $\mathrm{x}_i^j, y_i^j, j = 0,1,2,$ are the x, y coordinate of the three vertices in the ith images.

We can obtain EGF by the angle α_i between the line $\overrightarrow{\mathbf{O}_i\mathbf{A}_i}$ and the line $\overrightarrow{\mathbf{O}_i\mathbf{B}_i}$ to distinguish widths of the mouth with different expressions. And α_i can be calculated as,

$$\alpha_i = \arccos\left(\frac{\overrightarrow{\mathbf{O}_i\mathbf{A}_i} \bullet \overrightarrow{\mathbf{O}_i\mathbf{B}_i}}{\left|\overrightarrow{\mathbf{O}_i\mathbf{A}_i}\right|\left|\overrightarrow{\mathbf{O}_i\mathbf{B}_i}\right|}\right), \tag{3}$$

where α_i is angle between line $\overrightarrow{\mathbf{O}_i\mathbf{A}_i}$ and $\overrightarrow{\mathbf{O}_i\mathbf{B}_i}$, 'arccos' is the arc cosine transformation function, '$\bullet$' is the inner product of two vectors.

2). In Fig. 3, we obtain another EGF, which is denoted by the distance between upper lip and lower lip by four EGF points from images c and d. Because when you are happy or sad, your mouth is generally open or closed, respectively. The distance can be calculated by the difference of the coordinates of upper lip points and lower lip points.

3). Fig. 2 shows four examples of labeled training data, and each of them is labeled with 66 points around the mouth, the nose tip, eyes, and eyebrows. Using these 66 points can obtain a vector for planar image shapes,

$$\mathbf{x} = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n, \mathrm{y}_1, \mathrm{y}_2, \ldots, \mathrm{y}_n]^{\mathrm{T}}, \tag{4}$$

where (xi, yi) represents the ith landmark point coordinate, n is the total number of points (n=66). After the the searching procedure using ASM, we can obtain a set of new shape positions represented by (4). As a position feature, the shape vector is also used as an input to SVM with other EGF.

2.3 Overview of Support Vector Machine

The Support Vector Machine is used to do classification of two expressions in this paper. Here we describe SVM simply.

The SVM algorithm [18] needs to solve the following optimization problem,

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \text{,} \tag{5}$$

Subject to $$y_i(\mathbf{w}^T\phi(\mathbf{x}_i)+b) \geq 1-\xi_i\text{,}$$
$$\xi_i \geq 0, i = 1,\cdots,l.$$

where C is the term which penalizes the training error, b is the bias for the SVM, ξ_i is the ith slack variable vector, $\mathbf{x}_i \in R^n, i = 1,\cdots,l$ is the ith training vector, $y_i \in \{-1,1\}$, is the ith class label, and ϕ is a nonlinear mapping function by which training vector $\mathbf{x}_i$ is mapped into a higher dimensional space.

Using a kernel function, SVM projects training samples onto a high dimensional feature space where these samples can be divided linearly.

The decision function is given as,

$$\text{sgn}(\sum_{i=1}^{l} y_i\alpha_i K(\mathbf{x}_i,\mathbf{x})+b)\text{,} \tag{6}$$

here, α_i are the Lagrange multipliers of a dual optimization problem. Once (6) is obtained, classification of unseen test data is achieved.

The selection of a suitable kernel function is very important for the classification of SVM. The basic kernel is given as follows,

1). Radial Basis Function (RBF) kernel: $K(\mathbf{x}_i,\mathbf{x}_j) = \exp(-\gamma\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2), \gamma > 0.$

2). 'd' degree polynomial kernel: $K(\mathbf{x}_i,\mathbf{x}_j) = (\gamma\mathbf{x}_i{}^{\mathrm{T}}\mathbf{x}_j + r)^d, \gamma > 0.$

3). Sigmoid kernel: $K(\mathbf{x}_i,\mathbf{x}_j) = \tanh(\gamma\mathbf{x}_i{}^{\mathrm{T}}\mathbf{x}_j + r).$

3 Experimental Results and Discussions

In experiments, our method was applied to a Japanese Female Facial Expression (JAFFE) [19] database to evaluate the performance of classification of facial expressions. The database contains 213 images of 7 facial expressions including 6 basic facial expressions and 1 neutral posed by 10 Japanese female models. Each image has been rated on 6 emotion adjectives by 60 Japanese subjects [19].

Because we concentrate on the classification of happiness expression and sadness expression, 60 images including happiness and sadness of the JAFFE database are selected for training and testing. Thus, of the 60 images, 30 are happiness images and another 30 are sadness images, and 10 Japanese females are included for both expressions.

The Active Shape Model is generated by 55 images randomly selected from 60 images. The EGF acquired from the shape coordinates of facial expression feature points obtained from the fitting of ASM are used as an input to SVM. An RBF kernel is our

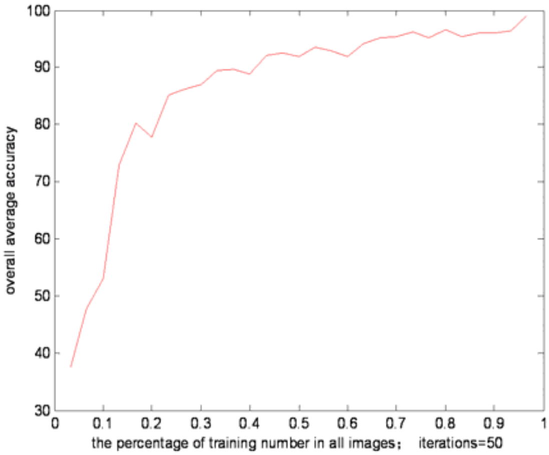


Fig. 4. The x-axis represents the number of training samples used as a ratio of the 60 total available samples for both expressions, and the y-axis represents the overall average accuracy for 50 iterations

first choice for the SVM system, and we use the cross-validation method to find the best parameters C and γ.

In the training and testing of SVM, we randomly select training samples from the set of 30 for each expression and vary the number of training samples from 1 to 29 for each expression. So the total ordinal training samples are 2, 4, … , 58. The testing is done on the remaining unused samples. The procedure is repeated for 50 iterations for each training sample size. We obtain the overall mean accuracy for each respective training sample size and plot the mean accuracy as a function of training sample size in Fig. 4. And this figure shows that the more training samples that are used, the higher the accuracy that can be obtained. From the figure, if we select 80% or even more as the training sets, the overall average accuracy is very high and does not have large fluctuation. So, we see the 80% as a very important turning point.

In experiments, ultimately 80% of samples for each class are used for training data, while the remaining samples form the test data. The overall mean accuracy using our proposed method is 97.3%. The happiness and sadness recognition rates are 98.5% and 96.1% respectively. Actually, 97.3% is one half of the sum of 98.5% and 96.1%.

In our experiments, we compare the classification performances of the proposed approach with other methods on the JAFFE database. For happiness expression recognition, the average accuracies in the papers [20], [21], [22] and [23] (six basic expressions are considered in these papers) are 50%, 50%, 30% and 70%, respectively, and the accuracy using our method is 98.5%. Meanwhile, for sadness expression recognition, the mean recognition rates in the above papers are 70%, 60%, 90%, and 60% respectively, and the recognition rate based on our method is 96.1%. In the paper [20], [21], their methods are based on the person-similarity weighted expression feature, but the similarities of persons obtained by face recognition algorithm are rough, thus the “true expression” feature can not be estimated accurately [20]. Nevertheless we extract distinguishable features, EGFs we propose, in terms of expression recognition in this paper. So we can get a relatively better result than theirs.

4 Conclusions

Experimental results based on the Japanese Female Facial Expression (JAFFE) database show that the approach to facial expression recognition for estimating patients' emotion we proposed in this paper, can classify happiness and sadness expressions with high accuracy. In this approach, we introduce a novel geometry feature, Emotion Geometry Feature (EGF), and Active Shape Model is used to extract the Emotion Geometry Feature points, and SVM ultimately classifies happiness and sadness expressions. We plan to extend happiness expression and sadness expression to six basic expressions further by adding more expression features to EGF.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No. 60671062 and 60571052) and National Basic Research Program of China (2005CB724303), and Shanghai Leading Academic Discipline Project (No. B112).

References

1. Aleksic, P.S., Katsaggelos, A.K.: Automatic facial expression recognition using facial animation parameters and multiStream HMMs. IEEE Trans. Inf. Forensics Secur. (2006)
2. Fasel, B., Luettin, J.: Automatic facial expression analysis: A survey. Pattern Recognition 36, 259–275 (2003)
3. Gholam Hosseini, H., Krechowec, Z.: Facial Expression Analysis for Estimating Patient's Emotional States in RPMS. In: Proc. of the 26th Annual International Conference of the IEEE, San Francisco, CA, USA, pp. 1517–1520 (2004)
4. Ekman, P., Friesen, W.V.: Emotion in the Human Face. Prentice-Hall, Englewood Cliffs (1975)
5. Fasel, B., Luettin, J.: Automatic Facial Expression analysis: a survey. Pattern Recognition 36, 259–275 (2003)
6. Pantic, M., Rothkrantz, L.L.M.: Automatic analysis of facial expressions: The state of the art. IEEE Trans. Pattern Ana. and Machine Intelligence 22, 1424–1455 (2000)
7. Guo, G., Dyer, C.R.: Learning from examples in the small sample case: Face expression recognition. IEEE Trans. Syst., Man, Cybern. B, Cybern. 35, 477–488 (2005)
8. Ma, L., Khorasani, K.: Facial expression recognition using constructive feedforward neural networks. IEEE Trans. Syst., Man, Cybern. B, Cybern. 34, 1588–1595 (2004)
9. Essa, I.A., Pentland, A.P.: Facial expression recognition using a dynamic model and motion energy. In: Int. Conf. Computer Vision, Cambrdige, MA (1995)
10. Bartlett, M.S., Littlewort, G., Braathen, B., Sejnowski, T.J., Movellan, J.R.: An approach to automatic analysis of spontaneous facial expressions. In: 5th IEEE Int. Conf. Automatic Face and Gesture Recognition, Washington, DC (2002)
11. Ekman, P., Friesen, W.V.: The facial action coding system: a technique for the measurement of facial movement. Consulting Psychologists Press, Inc., San Francisco (1978)

12. Pantic, M., Rothkrantz, L.J.M.: Facial action recognition for facial expression analysis from static face images. IEEE Trans. Sys., Man, Cybern. B: Cybern 34, 1449–1461 (2004)
13. Zhang, Y., Ji, Q.: Active and dynamic information fusion for facial expression understanding from image sequences. IEEE Trans. Pattern Analysis and Machine Intelligence 27, 699–714 (2005)
14. Cootes, T.: Introduction to active shape models, technical report (1998)
15. Kwo, W., Kin, L., Kit, N.: An accurate active shape model for facial feature extraction. In: Proc. Int. Sym. Int. Multimedia, Video and Speech, pp. 109–112 (2004)
16. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2001), `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
17. Hsu, C.-W., Chang, C.-C., Lin, C.-J.: A Practical Guide to Support Vector Classification (2008)
18. Vapnik, V.: Statistical learning theory. Wiley, New York (1998)
19. Lyons, M., Akamatsu, S., Kamachi, M., Gyoba, J.: Coding facial expressions with Gabor wavelets. In: 3rd International Conference on Automatic Face and Gesture Recognition, pp. 200–205 (1998)
20. Tan, H., Zhang, Y.-J.: Person-Similarity Weighted Feature for Expression Recognition. In: Yagi, Y., Kang, S.B., Kweon, I.S., Zha, H. (eds.) ACCV 2007, Part II. LNCS, vol. 4844, pp. 712–721. Springer, Heidelberg (2007)
21. Tan, H., Zhang, Y.: Person-Independent Expression Recognition Based on Person Similarity Weighted Distance. Jounal of Electronics and Information Technology 29, 455–459 (2007)
22. Wang, H., Ahuja, N.: Facial Expression Decomposition. In: ICCV, pp. 958–965 (2003)
23. Tian, Y., Kanade, T., Cohn, J.: Recognizing Action Units for Facial Expression Analysis. IEEE Trans. On PAMI 23, 97–115 (2001)
24. Cootes, T.F., Taylor, C.J., Cooper, D.H., Graham, J.: Active Shape Models- their training and application. Computer Vision and Image Understanding 61, 38–59 (1995)

A New Principal Axis Based Line Symmetry Measurement and Its Application to Clustering

Sanghamitra Bandyopadhyay and Sriparna Saha

Machine Intelligence Unit, Indian Statistical Institute, Kolkata, India-700108
{sanghami,sriparna_r}@isical.ac.in

Abstract. In this paper, at first a new line symmetry based distance is proposed which calculates the amount of symmetry of a point with respect to the first principal axis of a data set. The proposed distance uses a recently developed point symmetry based distance in its computation. Kd-tree based nearest neighbor search is used to reduce the complexity of computing the closest symmetric point. Thereafter an evolutionary clustering technique is described that uses this new principal axis based line symmetry distance for assignment of points to different clusters. The proposed GA with line symmetry distance based (GALS) clustering technique is able to detect any type of clusters, irrespective of their geometrical shape and overlapping nature, as long as they possess the characteristics of line symmetry. GALS is compared with the existing well-known GAK-means clustering algorithm. Three artificially generated and three real-life data sets are used to demonstrate its superiority.

1 Introduction

Partitioning a set of data points into some nonoverlapping clusters is an important topic in data analysis and pattern classification [1]. It has many applications, such as codebook design, data mining, image segmentation, data compression, etc. Many efficient clustering algorithms [1] have been developed for data sets of different distributions in the past several decades. Most of the existing clustering algorithms adopt the 2-norm distance measure in the clustering process.

In order to mathematically identify clusters in a data set, it is usually necessary to first define a measure of similarity or proximity which will establish a rule for assigning patterns to the domain of a particular cluster centroid. The measure of similarity is usually data dependent. It may be noted that one of the basic features of shapes and objects is symmetry. As symmetry is so common in the natural world, it can be assumed that some kind of symmetry exists in the clusters also. Based on this, a point symmetry based distance was developed in [2]. Kd-tree based nearest neighbor search is used to reduce the complexity of computing the point symmetry based distance. It is then used to develop a genetic algorithm based clustering technique, GAPS [2]. From the geometrical symmetry viewpoint, point symmetry and line symmetry are two widely discussed issues. Inspired by this, a line symmetry based distance was proposed in [3]. But the proposed distance had several drawbacks. The major shortcoming of the old line symmetry based distance was that its application is limited to two-dimensional data sets only. In this paper we have modified the line symmetry based distance proposed in

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 543–550, 2009.

[3]. The motivation of our present paper is as follows: to develop a new line symmetry based distance measure that removes the limitations of [3], and to incorporate it in a genetic clustering scheme that preserves the advantages of the previous GAPS clustering algorithm [2].

The proposed line symmetry based distance calculates the amount of symmetry of a point with respect to the first principal axis of the data points in a cluster. Principal component analysis (PCA) [4] is used to find the first principal axis of a dataset. The data set has a maximum amount of variation along the first principal axis. In the proposed clustering technique we have assigned a particular point to that cluster with respect to whose principal axis its line symmetry based distance is the minimum. Thus it can detect any shaped cluster which is symmetric with respect to its principal axis. *K*-means is a widely used clustering algorithm that has also been used in conjunction with the point-symmetry based distance measure in [5]. However *K*-means is known to get stuck at sub-optimal solutions depending on the choice of the initial cluster centers. In order to overcome this limitation, genetic algorithms have been used for solving the underlying optimization problem [6]. Genetic Algorithms (GAs) [7] are randomized search and optimization techniques guided by the principles of evolution and natural genetics, and having a large amount of implicit parallelism. GAs perform search in complex, large and multimodal landscapes, and provide near-optimal solutions for objective or fitness function of an optimization problem. In view of the advantages of the GA-based clustering method [6] over the standard *K*-means, the former has been used in this article. The proposed GA with line symmetry distance based clustering technique (GALS) is able to detect both convex and non-convex clusters of any shape and sizes as long as the clusters do have some line symmetry property. A Kd-tree based nearest neighbor search is utilized to reduce the computational complexity of computing the line symmetry based distance. The effectiveness of the proposed algorithm is demonstrated in identifying line symmetric clusters from three artificial and three real-life datasets. The clustering results are compared with those obtained by the well-known GAK-means clustering algorithm [6].

2 Newly Developed Line Symmetry Based Distance

Given a particular data set, we first find the first principal axis of this data set using Principal Component Analysis [4]. Let the eigen vector of the co-variance matrix of the data set with highest eigen value be $[eg_1\ eg_2\ eg_3\ eg_4 \dots\ eg_d]$, where d is the dimension of the original data. Then the first principal axis of the data set is given by:

$$\frac{(x_1 - c_1)}{eg_1} = \frac{(x_2 - c_2)}{eg_2} = \dots = \frac{(x_d - c_d)}{eg_d}$$

where the center of the data set is $\overline{c} = \{c_1, c_2, \dots, c_d\}$.

The obtained principal axis is treated as the symmetrical line of the relevant cluster, i.e., if the data set is indeed symmetrical then it should also be symmetric with respect to the first principal axis of the dataset identified by the principal component analysis (PCA). This symmetrical line is used to measure the amount of line symmetry of a particular point in that cluster. In order to measure the amount of line symmetry of a point ($\overline{x}$) with respect to a particular line i, $d_{ls}(\overline{x}, i)$, the following steps are followed.

1. For a particular data point $\overline{x}$, calculate the projected point $\overline{p}_i$ on the relevant symmetrical line i.
2. Find $d_{sym}(\overline{x},\overline{p}_i)$ as:

$$d_{sym}(\overline{x},\overline{p}_i) = \frac{\sum_{i=1}^{knear} d_i}{knear} \quad (1)$$

   where $knear$ unique nearest neighbors of $\overline{x}^* = 2 \times \overline{p}_i - \overline{x}$ are at Euclidean distances of d_is, $i = 1, 2, \ldots knear$. ANN (Approximate Nearest Neighbor) library [8] utilizing Kd-tree based nearest neighbor search is used to reduce the complexity of computing these d_is (as described in Section 2.1). Then the amount of line symmetry of a particular point $\overline{x}$ with respect to the symmetrical line of cluster i, is calculated as:

$$d_{ls}(\overline{x}, i) = d_{sym}(\overline{x},\overline{p}_i) \times d_e(\overline{x},\overline{c}) \quad (2)$$

   where $\overline{c}$ is the centroid of the particular cluster i and $d_e(\overline{x},\overline{c})$ is the Euclidean distance between the point $\overline{x}$ and $\overline{c}$.

It can be seen from Equation 1 that $knear$ cannot be chosen equal to 1, since if $\overline{x}^*$ exists in the data set then $d_{sym}(\overline{x},\overline{p}_i) = 0$ and hence there will be no impact of the Euclidean distance in the definition of $d_{ls}(\overline{x}, i)$. On the contrary, large values of $knear$ may not be suitable because it may underestimate the amount of symmetry of a point with respect to the first principal axis. Here $knear$ is chosen equal to 2. It may be noted that the proper value of $knear$ largely depends on the distribution of the data set. A fixed value of $knear$ may have many drawbacks. For instance, for very large clusters (with too many points), 2 neighbors may not be enough as it is very likely that a few neighbors would have a distance close to zero. On the other hand, clusters with too few points are more likely to be scattered, and the distance of the two neighbors may be too large. Thus a proper choice of *knear* is an important issue that needs to be addressed in the future.

It is evident that the symmetrical distance computation is very time consuming because it involves the computation of the nearest neighbors. Computation of $d_{ls}(\overline{x}, i)$ is of complexity $O(N)$. Hence for N points and K clusters, the complexity of computing the line symmetry based distance between all points to different clusters is $O(N^2K)$. In order to reduce the computational complexity, an approximate nearest neighbor search using the Kd-tree approach is adopted in this article.

2.1 Kd-tree Based Nearest Neighbor Computation

A K-dimensional tree, or Kd-tree is a space-partitioning data structure for organizing points in a K-dimensional space. A Kd-tree uses only those splitting planes those are perpendicular to one of the coordinate axes. ANN (Approximate Nearest Neighbor) is a library written in C++ [8], which supports data structures and algorithms for both exact and approximate nearest neighbor searching in arbitrarily high dimensions. In this article ANN library utilizing Kd-tree for nearest neighbor search is used to find d_is, where $i = 1, \ldots, knear$, in Equation 1 efficiently. Thus, it requires the construction of a Kd-tree consisting of N points in the data set, where N is the size of the data set. The construction of Kd-tree requires $O(NlogN)$ time and $O(N)$ space [9]. Friedman et al. [10] reported $O(logN)$ expected time for finding the nearest neighbor using Kd-tree.

3 GALS Clustering: Genetic Line Symmetry Distance Based Clustering Technique

In this section, a genetic clustering scheme, along the lines of the newly developed genetic clustering technique with point symmetry based distance (GAPS) [2], is proposed. Unlike GAPS, here the newly defined line symmetry based distance is used for cluster assignment and for calculation of fitness function. A brief overview of the basic steps of GALS are enumerated below. Given a particular value of the number of clusters, K, GALS partitions the data in K line symmetrical clusters.

3.1 String Representation and Population Initialization

The basic steps of GALS closely follow those of the conventional GA. Here center based encoding of the chromosome is used. Each string is a sequence of real numbers representing the *K* cluster centers and these are initialized to *K* randomly chosen points from the data set. This process is repeated for each of the $Popsize$ chromosomes in the population, where $Popsize$ is the size of the population. Thereafter five iterations of the *K*-means algorithm is executed with the set of centers encoded in each chromosome. The resultant centers are used to replace the centers in the corresponding chromosomes. This makes the centers separated initially.

3.2 Fitness Computation

In order to compute the fitness of the chromosomes, the following steps are executed.
•Find the first principal axis of each cluster using principal component analysis [4]. This first principle axis is treated as the symmetrical line for each cluster.
• For each data point $\overline{x}_i$, $i = 1, \ldots N$, where N is the total number of points present in the data set, calculate the projected point $\overline{p}_{ki}$ on the first principal axis of cluster C_k, $k = 1, \ldots K$, where K is the total number of clusters. Then compute $d_{ls}(\overline{x}_i, k)$ using Equation 2.
• The point $\overline{x}_i$ is assigned to cluster k iff $d_{ls}(\overline{x}_i, k) \leq d_{ls}(\overline{x}_i, j),\ j = 1, \ldots, K,\ j \neq k$ and $d_{sym}(\overline{x}_i, \overline{p}_{ki}) \leq \theta$. For $d_{sym}(\overline{x}_i, \overline{p}_{ki}) > \theta$, point $\overline{x}_i$ is assigned to some cluster m iff $d_e(\overline{x}_i, \overline{c}_m) \leq d_e(\overline{x}_i, \overline{c}_j),\ j = 1, 2 \ldots K,\ j \neq m$. In other words, point $\overline{x}_i$ is assigned to that cluster with respect to whose principal axis its LS-distance is the minimum, provided the total "symmetricity" with respect to it is less than some threshold θ. Otherwise assignment is done based on the minimum Euclidean distance criterion as normally used in [6] or the *K*-means algorithm.

The value of θ is kept equal to the maximum nearest neighbor distance among all the points in the data set [2].

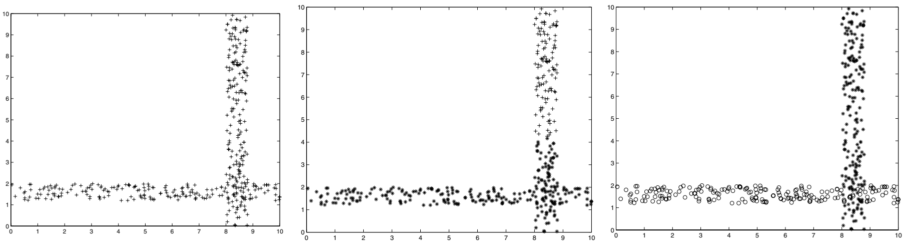
After the assignments are done, the cluster centres encoded in the chromosome are replaced by the mean points of the respective clusters. Subsequently for each chromosome *clustering_metric,M*, is calculated as: $M = \sum_{i=1}^{K} \sum_{j=1}^{n_i} d_{ls}(\overline{x}_i^j, i)$, where n_i denotes the number of points in ith cluster, and $\overline{x}_i^j$ denotes the jth point of the ith cluster. Then the fitness function of that chromosome, fit, is defined as the inverse of M, i.e., $fit = \frac{1}{M}$. This fitness function, fit, will be maximized by using genetic algorithm. (Note that there could be other ways of defining the fitness function).

3.3 Genetic Operators

Roulette wheel selection is used to implement the proportional selection strategy. Here, we have used the normal single point crossover [7]. Each chromosome undergoes mutation with a probability μ_m. We have used the mutation operation similar to that used in GA based clustering [6]. In GALS, the processes of fitness computation, selection, crossover, and mutation are executed for a maximum number of generations. The best string seen upto the last generation provides the solution to the clustering problem. Elitism has been implemented at each generation by preserving the best string seen upto that generation in a location outside the population. Thus on termination, this location contains the centers of the final clusters.

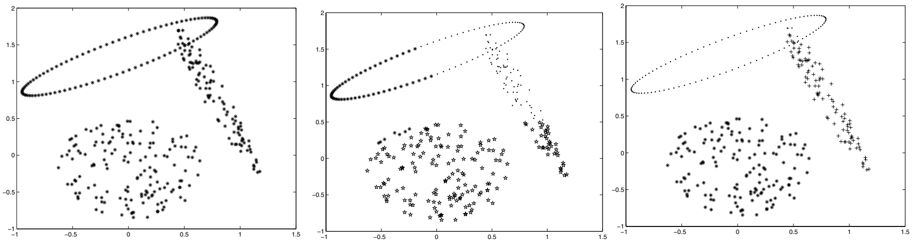
4 Implementation Results

The experimental results comparing the performances of GALS and GAK-means clustering algorithms are provided for three artificial and three real-life data sets. For the newly developed GALS clustering, a value of θ is determined from the data set as discussed in Section 3.2. For both the genetic clustering techniques, GAK-means and GALS, the following parameter values are kept: population size=100, number of generations=30, probability of crossover = 0.8, probability of mutation=0.1. Increasing the number of generations did not improve the performance of any of these algorithms.



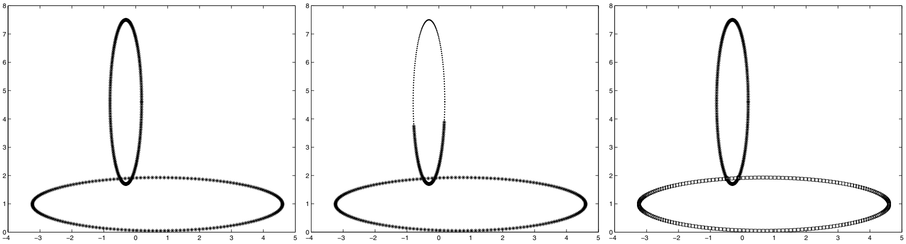
(a) (b) (c)

Fig. 1. (a) *Line_2_2* (b) Partitioning obtained by GAK-means algorithm for $K = 2$ (c) Partitioning obtained by GALS clustering algorithm for $K = 2$



(a) (b) (c)

Fig. 2. (a) *Mixed_3_2* (b) Partitioning obtained by GAK-means algorithm for $K = 3$ (c) Partitioning obtained by GALS clustering algorithm for $K = 3$



(a) (b) (c)

Fig. 3. (a) *Ring_2_2* (b) Partitioning obtained by GAK-means algorithm for $K = 2$ (c) Partitioning obtained by GALS clustering algorithm for $K = 2$

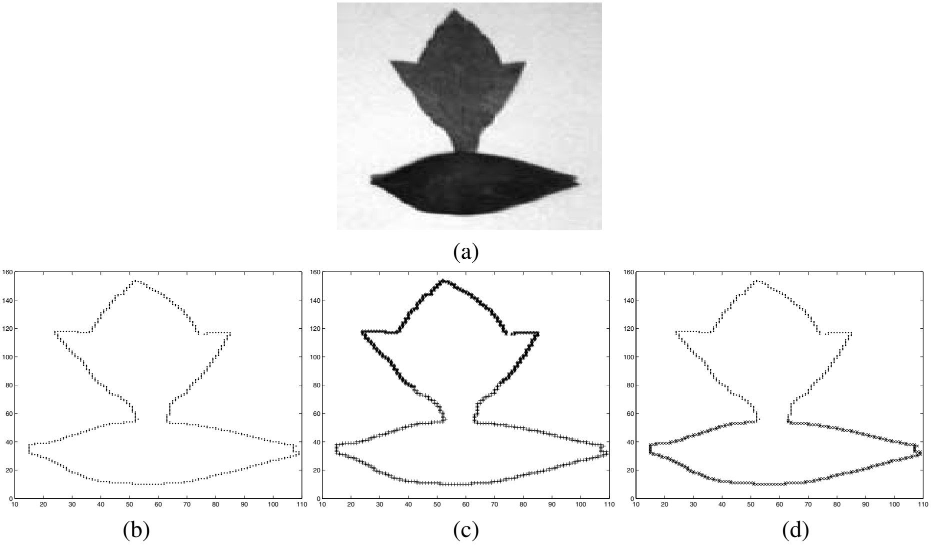
• *Line_2_2*: This data set, used in [2], consists of two bands as shown in Figure 1(a), where each band consists of 200 data points. The final clustering results obtained by GAK-means and GALS clustering techniques are provided in Figures 1(b) and 1(c), respectively. As expected GAK-means performs poorly for this data since the clusters are not hyperspherical in nature. Our proposed GALS is able to detect the proper partitioning from this data set as the clusters possess the line symmetry property.

• *Mixed_3_2*: This data set, used in [11] is a combination of ring-shaped, compact and linear clusters shown in Figure 2(a). The total number of points in it is 350. The final clustering results obtained after application of GAK-means and GALS are shown in Figures 2(b), and 2(c), respectively, where GAK-means is found to fail in providing the proper partitioning. The proposed GALS is able to detect the proper partitioning.

• *Ring_2_2*: This data set is distributed on two crossed ellipsoidal shells, shown in Figure 3(a). This is a non-convex symmetrical data set, used in [2]. The final clustering results corresponding to GAK-means and GALS are shown in Figures 3(b) and 3(c), respectively. GAK-means again fails here in detecting ellipsoidal shaped clusters. But as the clusters present here are line symmetric, the proposed GALS is able to detect the clusters well.

•*Two_leaves1*: Most of the natural scenes, such as leaves of plants, have the line symmetry property. Figure 4(a) shows the two real leaves of *Ficus microcapa* and they overlap a little each other. First the sobel edge detector [12] is used to obtain the edge pixels in the input data points which is shown in Figure 4(b). The obtained partitionings after execution of both GAK-means and GALS clustering techniques are shown in Figures 4(c) and 4(d), respectively. The proposed GALS again demonstrates a satisfactory clustering result.

•*Iris*: This data set, obtained from [13], represents different categories of irises characterized by four feature values [14]. It has three classes: Setosa, Versicolor and Virginica. It is known that the two classes (Versicolor and Virginica) have a large amount of overlap while the class Setosa is linearly separable from the other two. As this is a higher dimensional data set, no visualization is possible. In order to measure the segmentation solution quantitatively, we have also calculated *Minkowski Score*(MS) [15]. This is a measure of the quality of a solution given the true clustering. For MS, the optimum score is 0, with lower scores being “better”. The proposed GALS clustering technique


(a)

(b)

(c)

(d)

Fig. 4. (a) *Two_leaves1* data (b) Edge pixels of leaves as input data points (c) Partitioning obtained by GAK-means for $K = 2$ (d) Partitioning obtained by GALS for $K = 2$

obtains a MS score of 0.603745 ± 0.02 for this data set while the MS score corresponding to the partitioning provided by GAK-means algorithm is 0.6256 ± 0.013.

• *Cancer*: Here we use the Wisconsin Breast Cancer data set, obtained from [13]. Each pattern has nine features corresponding to clump thickness, cell size uniformity, cell shape uniformity, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli and mitoses. There are two categories in the data: malignant and benign. The two classes are known to be linearly separable. Again for this data set the performance of GALS-clustering is slightly better than that of GAK-means in terms of mean MS. GALS clustering technique attains MS-score of 0.353192 ± 0.011 for this data set while GAK-means attains MS score of 0.367056 ± 0.024.

5 Discussion and Conclusion

In this paper a new line symmetry based distance is proposed which measures the total amount of symmetry of a point with respect to the first principal axis of a cluster. Kd-tree based nearest neighbor search is used to reduce the complexity of symmetry based distance computation. A genetic clustering technique (GALS) is also proposed here that incorporates the new line symmetry based distance while performing cluster assignments of the points and in the fitness computation. The major advantages of GALS are as follows. Like GAK-means, use of GA enables the algorithm to come out of local optima, making it less sensitive to the choice of the initial cluster centers. The proposed clustering technique can cluster data sets with the property of line symmetry successfully. The effectiveness of the proposed algorithm is demonstrated in detecting clusters having line symmetry property from three artificial and three real-life data sets.

Other than the clustering experiments using the leaf example, it is an interesting future research topic to extend the results of this paper to face recognition.

References

1. Everitt, B.S., Landau, S., Leese, M.: Cluster Analysis. Arnold, London (2001)
2. Bandyopadhyay, S., Saha, S.: GAPS: A clustering method using a new point symmetry based distance measure. Pattern Recognition 40, 3430–3451 (2007)
3. Saha, S., Bandyopadhyay, S.: A genetic clustering technique using a new line symmetry based distance measure. In: Proceedings of Fifth International Conference on Advanced Computing and Communications (ADCOM 2007), Guwahati, India, pp. 365–370. IEEE Computer Society Press, Los Alamitos (2007)
4. Jolliffe, I.: Principal Component Analysis. Springer Series in Statistics, England (1986)
5. Su, M.C., Chou, C.H.: A modified version of the k-means algorithm with a distance based on cluster symmetry. IEEE Transactions Pattern Analysis and Machine Intelligence 23(6), 674–680 (2001)
6. Maulik, U., Bandyopadhyay, S.: Genetic algorithm based clustering technique. Pattern Recognition 33, 1455–1465 (2000)
7. Holland, J.H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, AnnArbor (1975)
8. Mount, D.M., Arya, S.: ANN: A library for approximate nearest neighbor searching (2005), http://www.cs.umd.edu/~mount/ANN
9. Anderberg, M.R.: Computational Geometry: Algorithms and Applications. Springer, Heidelberg (2000)
10. Friedman, J.H., Bently, J.L., Finkel, R.A.: An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software 3(3), 209–226 (1977)
11. Bandyopadhyay, S., Saha, S.: A point symmetry based clustering technique for automatic evolution of clusters. IEEE Transactions on Knowledge and Data Engineering 20(11), 1–17 (2008)
12. Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Addison-Wesley, Massachusetts (1992)
13. http://www.ics.uci.edu/~mlearn/MLRepository.html
14. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annals of Eugenics 3, 179–188 (1936)
15. Ben-Hur, A., Guyon, I.: Detecting Stable Clusters using Principal Component Analysis in Methods in Molecular Biology. Humana press (2003)

Class-Dependent Feature Selection for Face Recognition

Zhou Nina and Lipo Wang

School of Electrical and Electronic Engineering
Nanyang Technological University
Block S1, 50 Nanyang Avenue, Singapore 639798

Abstract. Feature extraction and feature selection are very important steps for face recognition. In this paper, we propose to use a class-dependent feature selection method to select different feature subsets for different classes after using principal component analysis to extract important information from face images. We then use the support vector machine (SVM) for classification. The experimental result shows that class-dependent feature selection can produce better classification accuracy with fewer features, compared with using the class-independent feature selection method.

1 Background

Automatic face recognition has experienced a relatively long process from the 1960s until now. In the 1960s, the first semi-automated face recognition system was developed [1]. The recognition process of this system included the location of features like eyebrows, eyes, noses and so on, calculation of distances and ratios to a common reference point and template matching. Later in the 1970s, some subjective features like hair color and lip thickness were developed by Goldstein et al. [2] to automate the recognition system. However, these two early solutions have one drawback, namely manually computing the measurements and locations. In order to deal with this problem, in the 1980s, Kirby and Sirovich [3] applied principal component analysis (PCA) to extract important information by singling out important face features. This application was thought of the first successful example of automatic face recognition systems. Following that, more feature extraction techniques like independent component analysis (ICA) [4] and linear discriminant analysis (LDA) [5], [6] were proposed.

Face recognition has three basic sequential processes: preprocessing, feature extraction or selection, and recognition. For different images or databases, preprocessing can vary from noise removal, normalization, to space transformation, e.g., Fourier transformation [7]. Since a large amount of information is stored in images and it is impractical to use all information in computation, feature extraction or feature selection [8], [9] is very necessary. Various feature extraction techniques, for example, PCA [10], [11], [12] and its variants [13], [5], fisher linear discriminant analysis (FLDA) [5], [6], [7], general tensor discriminant analysis

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 551–558, 2009.

(GTDA) [14], and ICA [4], have been used to extract face features [7]. Feature selection is another way of obtaining compact face information. This involves first determining some characteristics of faces, such as the distance between eyes, the width of nose, and the length of jaw line [15], [16], then combining all those information to form a feature vector. Although features obtained by this method can be easier to interpret than those by feature extraction, it is not cost-effective work to determine which features are desirable for face images and then compute those features. In this case, feature extraction is more reliable to obtain fully representative information of face images, which will be our first step in dimensionality reduction. Recognition of faces also has many kinds of techniques, such as template matching [1], [11] and various classifiers [17], [18], [19], [20], [21].

Considering the possibility that different features have different classification power for different classes, in this paper, we propose to adopt the class-dependent feature selection [22], [23], [24] method for further dimensionality reduction in the second step. Class-dependent feature selection chooses a feature subset for each class, i.e., different feature subsets for different classes. The usual class-independent feature selection method chooses a common feature vector for all classes. This is an novel application which is not to show that our method outperforms all other existing methods on classification performance, but intends to show that class-dependent feature selection is better than class-independent feature selection. Therefore, this application will provide the possibility to employ the idea of class-dependent feature selection with many other feature extraction methods.

This paper is organized as follows. In section 2, we briefly review PCA and then describe the class-dependent feature selection method. In section 3, we utilize the SVM to realize the classification on the ORL data set [25] and compare results of class-dependent feature selection method with those of class-independent feature selection method, and also some published results. In section 4, we make a discussion about the present work.

2 Methodology

As an efficient dimensionality reduction technique in data analysis and pattern recognition, PCA [10], [13], [11], [7], [26] has already been widely used in face recognition systems. PCA [27], [10], [12] computes principal components of images, thereby transforming training images (denoted as matrix X with N samples in rows and p features in columns) into a new space of the principal components. The basic steps are: (1) calculating the covariance matrix of data matrix X; (2) determining eigenvalues and eigenvectors of this covariance matrix; (3) selecting m ($m < p$) significant eigenvectors to form transformation matrix T with the first row corresponding to the most important eigenvectors; and (4) obtaining the projected images by calculating $Y^T = TX^T$, here Y is a matrix with N rows and m columns. Through PCA, the dimension of original images is reduced to m . The dimension of kept components decides the amount of information lost.

After feature extraction, we propose to select class-dependent features from obtained principal components. For the case of high-dimensional features, it is impractical for us to sequentially add all features and evaluate all feature subsets. Since in this paper we will adopt the ORL face database [25] which has a very high dimension in our experiment, each time we will add 3 features into the previous feature subset and stop at a predetermined threshold of the dimension, e.g., 30. Each feature subset is evaluated by the SVM and the feature subset with the highest classification accuracy is chosen as the superior one of the current class.

The class-dependent feature selection is described as follows:

1. Based on the strategy of "one-against-all" [28], [29], we convert a multi-class problem into several two-class problems. For example, the problem of classifying face images is converted into 2-class classification problems, where each problem only includes two classes, i.e., one being the original class and the other one consisting of all the other classes.
2. For each 2-class problem, we adopt the class separability measure (CSM) [30], [29] to evaluate features' ranking for each class. The CSM evaluates how well two classes are separated by a feature vector. The greater the distance between different classes, the easier the classification task. For example, if S_w denotes the within-class distance and S_b denotes the between-class distance, the ratio S_w/S_b can be used to measure the separability of the classes. The smaller the ratio, the better the separability. The importance of a feature may be evaluated by ratio S_w/S_b calculated *after the feature is removed from the data set*, i.e., S'_w/S'_b. The greater S'_w/S'_b is, the more important the removed attribute is. Hence we may evaluate the importance level of the attributes according to the ratio S_w/S_b with an attribute deleted each time in turn. Each class will have a feature importance ranking list. For example, for problem 1, its ranking list of features measures the importance of features in classifying class 1 from the other classes. Therefore, this feature importance ranking list is specific to class 1. Likewise for class 2, class 3,..., and class C.

$$S_w = \sum_{c=1}^{C} P_c \sum_{j=1}^{n_c} \left[(\mathbf{X}_{cj} - \overline{m}_c)(\mathbf{X}_{cj} - \overline{m}_c)^T \right]^{1/2} \tag{1}$$

$$S_b = \sum_{c=1}^{C} P_c \left[(\overline{m}_c - \overline{m})(\overline{m}_c - \overline{m})^T \right]^{1/2} \tag{2}$$

   Here P_c is the probability of the c-th class, and n_c is the number of samples in the c-th class. $\mathbf{X}_{cj}$ is the j-th sample in the c-th class, $\overline{m}_c$ is the mean vector of the c-th class, and $\overline{m}$ is the mean vector of all samples in the data set.
3. According to feature importance ranking lists obtained in step 2, we need to determine a feature subset for each class. We can choose a classifier, e.g., the

SVM, to evaluate feature subsets and determine the most contributive one. For each class, each feature subset is formed by sequentially adding one or several features into the previous subset. The feature subset with the highest classification accuracy is chosen as the most contributive one. Usually, we ranked all d features and formed all d feature subsets as follows. The top 1 feature consists of the first feature subset. The second feature subset consists of the top two ranked features. The d-th feature subset includes all features. Whereas, in practical situation, e.g., for the case of high-dimensional features, it will be computationally expensive for us to sequentially form all feature subsets and evaluate them. In this paper, we will adopt ORL face database [25] which has a very high dimension in our experiment. Each time we will try add 3 features into the previous feature subset and stop at a predetermined threshold of the dimension, e.g., 30. We also evaluated all 30 feature subsets, i.e., top first, top first and second feature, ..., top 30 features and found that accuracies are increasing. When more features are added, the accuracies of the formed feature subsets keep stable. Each feature subset is evaluated by the SVM and the feature subset with the highest classification accuracy is chosen as the superior one of the current class.

In order to conveniently describe the class-dependent feature subset, we attempt to use a feature mask to express the state of each feature. The feature mask only has two elements '0' and '1', in which '0' represents the absence of a particular feature and '1' represents the presence of the feature. For example, considering a data set with 5 features $\{x_1, x_2, x_3, x_4, x_5\}$, if the optimal feature subset obtained is with the second and forth features deleted, the feature mask for this feature subset should be {1, 0, 1, 0, 1}.

After selecting class-dependent feature subsets, we adopt the SVM with RBF kernel [30], [29] for the classification because of many advantages of the SVM, such as fast speed, high recognition rate. Since class-dependent features can not directly be input into the original SVM, we adopt the class-dependent SVM classifier as [29] described. Based on the class-dependent feature mask, we construct a classifier model for each class, i.e., forming class-dependent models. Each model is trained using feature subsets specific to the corresponding class. Each testing data is filtered by the feature mask of the corresponding class before input into one model. The maximum value of all models' outputs determines the class of the testing data.

3 Experiments

3.1 Data Description and Preprocessing

In this experiment, we selected the Cambridge ORL face database [25] as our experimental data. It includes 40 subjects (faces or classes), each of which has 10 slightly different face images. Therefore, the total number of face images is 400. Each face image is in gray scale and has 112 by 92 pixels. For processing conveniently, we reshaped each original image matrix into a column vector, which has the dimension 10304. All image vectors constitute a new image matrix with 400 samples in rows and 10304 features in columns. Considering the effect of illumination on different face images, we adopted the normalization as [7] described, i.e., subtracting the mean of each image and divide by the variance. After that, all images were equally distributed in terms of energy [7].

3.2 Implementation and Results

Since normalized face images have a high dimension, PCA is adopted to extract principal components for each face image. The dimension of the original face image is reduced to 399 by PCA. The class-dependent feature selection method is used to further reduce features for each class. Before feature selection, we separate all 400 images faces into 200 training images and 200 testing images. For the training set, we calculate features' importance ranking for each class. Although principal components (i.e., features obtained from PCA) are sorted according to another importance measure, i.e., the eigenvalue, we believe that features' ranking for different classes are different. In Table 1, we provide the number of features selected for each class. We can see that class-dependent feature selection method selected rather different features for different classes, i.e., as many as 8 features for several classes and as few as 1 features for other classes (See Table 1). When combining features subsets of all classes in Table 1, we include 26 different features in the union set. Through the SVM, we obtained different feature masks for different classes. For example, in Table 2, we provide feature masks of the first 5 classes. For classification, the 200 testing samples were processed in the same way as the training set and tested on a class-dependent SVM classifier to produce a 98% classification accuracy, which is better than the classification result by the class-independent feature selection method (see Table 3). Table 3 provides classification results of different number of features (principal components) by using normal class-independent feature selection method. The class-independent feature selection method selected 30 features to produce classification accuracy 97.5%. Finally we compared our result with that of some existing methods in Table 4. All those experiments were done on the ORL database but with different recognition methods. The result shows that our method is very comparable.

4 Summary and Discussion

In this paper, we applied the class-dependent feature selection methods [29] on face recognition problems, after processing the whole data set using PCA. Although many fully-fledged feature extraction techniques like PCA, LDA and elastic bunch graph matching [31] exist and have good successful experiences in face recognition, we still proposed this novel application on face recognition to detect if our proposed class-dependent method has advantages in classification performance over the class-independent feature selection method. In the process of experiments, PCA was used as a preprocessing step to extract face features. Then both class-dependent and class-independent feature selection were used to

Table 1. Number of features selected for each classes

Class No.	1	2	3	4	5	6	7	8	9	10
Number of features	2	3	4	7	3	1	3	2	3	2
Class No.	11	12	13	14	15	16	17	18	19	20
Number of features	3	3	1	2	4	5	3	2	1	5
Class no.	21	22	23	24	25	26	27	28	29	30
Number of features	2	2	3	1	6	3	2	8	2	1
Class no.	31	32	33	34	35	36	37	38	39	40
Number of features	2	5	3	2	6	8	5	4	3	8

Table 2. Feature masks of the first 5 classes

Class 1	1 1 0 0 0 0 0 0
Class 2	1 1 1 0 0 0 0 0
Class 3	1 1 1 1 0 0 0 0
Class 4	1 1 1 1 1 1 1 0
Class 5	1 1 1 0 0 0 0 0

Table 3. Classification accuracy for variant numbers of features using class-independent feature selection method (10 fold cross validation)

Number of components (features)	5	10	15	20	25	30
Accuracy	80.50%	87.50%	92.00%	94.75%	95.25%	97.50%

Table 4. Comparisons of classification accuracies with existing methods

Method	Proposed	Class-independent after PCA	Kim et al.'s [13]	Lu et al's [6]
Classifier	SVM with RBF kernel	SVM with RBF kernel	Linear SVM	Nearest Neighbor
Number of selected features	26	30	120	22
Accuracy	98.0%	97.5%	97.5%	96.0%

select features for recognition. The classification results showed that the introduction of our proposed method selected features specific for each class (face). Therefore, the feature dimension of each class is less than that produced by the class-independent feature selection. Also a better classification performance is obtained compared to normal class-independent feature selection methods. Besides, the corresponding face recognition system after class-dependent feature selection has one more advantage over the normal face recognition system: when adding one more class (face) into the existing system, class-dependent recognition system does not need to re-train whole system [22]. However, the recognition system based on class-independent feature selection needs to re-train the whole system if one more class is added.

Noticeably the class-dependent feature selection method is likely to be more computationally expensive than other conventional feature selection methods. However, the extra computational cost may be worthwhile in certain applications where improvements of accuracy or reduction of data dimensionality are very important and meaningful.

References

[1] Li, S.Z., Jain, A.K.: Handbooks of face recognition. Springer, New York (2001)

[2] Goldstein, A.J., Harmon, L.D., Lesk, A.B.: Identification of Human Faces. Proceedings of the IEEE 59, 748–760 (1971)

[3] Sirovich, L., Kirby, M.: Low-dimensional procedure for the characterization of human faces. Journal of the Optical Society of America 3(4), 519–524 (1987)

[4] Liu, C.J., Wechsler, H.: Independent component analysis of Gabor features for face recognition. IEEE Trans. on Neural Networks 14(4), 919–928 (2003)

[5] Liu, C.J.: Gabor-based kernel PCA with fractional power polynomial models for face recognition. IEEE Trans. on Pattern Analysis and Machine Intelligence 26(5), 572–581 (2004)

[6] Lu, J.W., Plataniotis, K.N., Venetsanopoulos, A.N.: Face recognition using LDA-Based algorithms. IEEE Trans. on Neural Networks 14(1), 195–200 (2003)

[7] Vishnubhotla, S.: Face recognition. Project Work (2005)

[8] Duda, R., Hart, P.: Pattern Classification and Scene Analysis. Wiley, New York (1973)

[9] Fukunaga, K.: Statistical Pattern Recognition. Academic Press, New York (1989)

[10] Joo, S.W.: Face recognition using PCA and FDA with intensity normalization, Project Work (2003), http://www.cs.umd.edu/~swjoo/reports/739Q_report.pdf

[11] Perlibakas, V.: Distance measures for PCA-based face recognition. Pattern Recognition Letters 25(6), 711–724 (2004)

[12] Wang, L.P., Fu, X.J.: Data Mining with Computational Intelligence. Springer, Berlin (2005)

[13] Kim, K.I., Jung, K.C., Kim, H.J.: Face Recognition Using Kernel Principal Component Analysis. IEEE Signal Processing Letters 9(2), 40–42 (2002)

[14] Tao, D.C., Li, X.L., Wu, X.D., Maybank, S.J.: General Tensor Discriminant Analysis and Gabor Features for Gait Recognition. IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI) 29(10), 1700–1715 (2007)

[15] Craw, I., Tock, D., Bennett, A.: Finding face features. In: Sandini, G. (ed.) ECCV 1992. LNCS, vol. 588, pp. 92–96. Springer, Heidelberg (1992)

[16] Johnson, R., Bonsor, K.: How facial recognition systems work, http://computer.howstuffworks.com/facial-recognition.htm

[17] Liu, Q.S., Lu, H.Q., Ma, S.D.: A non-parameter Bayesian classifier for face recognition. Journal of Electronics 20(5), 362–370 (2003)

[18] Liu, S., Wang, Z.: A face recognition classifier based on the RBPNN model. Computer Engineering and Science (2) (2006)

[19] Zhang, Y.K., Liu, C.Q.: Face recognition based on support vector machines and nearest neighbor classifier. Journal of Systems Engineering and Electronics (3) (2003)

[20] Zhuang, L., Ai, H.Z., Xu, G.Y.: Training support vector machines for video based face recognition. In: The 2nd International Conference on Image and Graphics, vol. 4875, pp. 737–743 (2002)

[21] Zhuang, L., Ai, H.Z., Xu, G.Y.: Video based face recognition by support vector machines. In: International Conference on Computer Vision, Pattern Recognition and Image Processing, Durham, North Carolina, USA (2002)

[22] Baggenstoss, P.: Class-specific feature sets in classification. In: Proceedings of IEEE International Symposium on Intelligent Control (ISIC), pp. 413–416 (1998)

[23] Oh, I.S., Lee, J.S., Suen, C.Y.: Analysis of class separation and combination of class-dependent features for handwriting recognition. IEEE Trans. on Pattern Analysis and Machine Intelligence 21(10), 1089–1094 (1999)

[24] Zhou, N.N., Wang, L.P.: A novel support vector machine with class-dependent features for biomedical data. In: Verma, B., Blumenstein, M. (eds.) Pattern Recognition Technologies and Applications: Recent Advances, Hershey, New York, Information Science Reference, pp. 284–298 (2008)

[25] AT & T Laboratories Cambridge: The ORL Databases of Faces (1992-1994), http://www.cl.cam.ac.uk/Research/DTG/attarchive:pub/data/att_faces.zip

[26] Zhao, H.T., Yuen, P.C., Kwok, J.T.: A novel incremental principal component analysis and its application for face recognition. IEEE Trans. on Systems, Man and Cyber. 36(4), 873–886 (2006)

[27] Belhumeur, P.N., Hespanha, J.P., Kreigman, D.J.: Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection. IEEE Trans. on Pattern Analysis and machine intelligence 19(7), 711–720 (1997)

[28] Hsu, C.W., Lin, C.J.: A Comparison of methods for multi-class support vector machines. IEEE Trans. on Neural Network 13(2) (2002)

[29] Wang, L.P., Zhou, N.N., Chu, F.: A general wrapper approach to selection of class-dependent features. IEEE. Trans. on Neural networks 19(7), 1267–1278 (2008)

[30] Fu, X.J., Wang, L.P.: Data dimensionality reduction with application to simplifying RBF network structure and improving classification performance. IEEE Trans. on System, Man, and Cyber.–Part B: Cyber. 33(3), 399–400 (2003)

[31] Wiskott, L., Fellous, J.-M., Kuiger, N., von der Malsburg, C.: Face recognition by Elastic Bunch Graph Matching. IEEE Trans. on Pattern Analysis and Machine Intelligence 19(7), 775–779 (1997)

Partial Clustering for Tissue Segmentation in MRI

Nicolau Gonçalves, Janne Nikkilä, and Ricardo Vigário

Adaptive Informatics Research Centre
Helsinki University of Technology
P.O. Box 5400, FIN-02015, Finland
firstname.lastname@tkk.fi

Abstract. Magnetic resonance imaging (MRI) is a imaging and diagnostic tool widely used, with excellent spatial resolution, and efficient in distinguishing between soft tissues. Here, we present a method for semi-automatic identification of brain tissues in MRI, based on a combination of machine learning approaches. Our approach uses self-organising maps (SOMs) for voxel labelling, which are used to seed the discriminative clustering (DC) classification algorithm. This method reduces the intensive need for a specialist, and allows for a rather systematic follow-up of the evolution of brain lesions, or their treatment.

1 Introduction

Magnetic resonance imaging (MRI) is a widely used clinical imaging technique. Advantages over other methods include its non-invasiveness, the ability to distinguish between soft tissues, and an excellent spatial resolution [1]. Depending on the type of evaluation required [2,3], a number of possible acquisition setups can be used. In most clinical applications, several such sequences are acquired.

Here, we propose a technique for MRI semiautomatic analysis. The goal is to improve tissue segmentation, allowing for an easier detection and follow up of brain pathologies. This method should reduce the workload of medical doctors, as well as enable systematic screening of large groups of patients. Simple human interaction is used to select relevant clusters.

In [4] we observed that a reliable use of independent component analysis (ICA, [5]), used as a pre-processing technique, improved the analysis of MR images. Due to a clear increase in tissue contrast, the components offer themselves, by simple visual inspection, valuable segmentation information. Subsequent segmentation was then based on self-organising maps (SOM, [6]).

This study builds on previous results, in a more systematic way of classifying brain tissues and moreover, the ability to obtain partial volume information (PVI). Emphasis is put on analysing degenerative diseases, such as multiple sclerosis (MS).

We used discriminative clustering (DC, [7]) to perform the classification of the brain tissues. Since DC is a supervised method, it requires labelled training

[1] We acknowledge Aarto Klami for his support in the DC implementation.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 559–566, 2009.

data. Yet, the targeted classification task was unsupervised, i.e., no labelled data was available. A set of labelled data was then produced based on a boosting combination of ICA and SOM, following the procedure in [4].

The method was evaluated in a simulated brain, to obtain a quantitative performance evaluation. The results are then compared to the ones obtained in a recent voxel classification method proposed by Tohka et al. [8]. The method from Tohka et al. uses finite mixture models to do brain image classification, where the parametric form of a tissue *pdf* is assumed to be known.

Our basic assumption is that each image can be considered as a linear combination of contributions from each tissue present. Although not strictly correct, this assumption has proven to be a rather good first approximation.

2 Data

We used two sets of simulated MRI from the BrainWeb database [9,10], with 1 mm^3 of spatial resolution and no intensity nonuniformity. The first set, henceforth called *normal-set*, contained T1, T2 and proton density (PD) sequence images, with 3% of noise. The only tissues retained for analysis were white- and grey-matter and cerebrospinal fluid (CSF). The second set (*lesion-set*, see Fig. 1) was generated using the noiseless moderate MS phantom. It consisted of 13 images with various echo and repetition times, TE and TR respectively. The time of echo (TE) and repetition (TR) were chosen so that the typically used T1, T2 and PD images were included [11]. The remaining 10 images were obtained so that (TE,TR) were distributed between T1 and PD (3 images) and PD and T2 (7 images).

3 Methods

With the objective of obtaining the PVI of the different voxels, a method providing voxel memberships to different classes is needed. Furthermore, since MS is a degenerative disease, it is valuable to evaluate its various stages, as well as the directions of progression of the disease. Discriminative clustering is perfectly suited for this task. To estimate the needed DC label information, we run a consistency study based on SOM. To obtain better results, we also denoised the SOM input data with ICA. In the following, we will review each step of the analysis method.

3.1 Independent Component Analysis on the Innovation

We assume the observation vector, $\boldsymbol{x}$, to be generated as a linear combination of statistically independent sources $\boldsymbol{s}$, *i.e.*, $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}$. ICA estimates $\boldsymbol{s}$ by optimising a number of possible contrast functions, typically based on high-order statistics [5].

ICA on the innovations was derived in [12] to deal with time-dependent stochastic processes. Innovation is defined as the error between a stochastic process and its best prediction given its past,*i.e.* it contains all the process information not explained by a predictive model. The benefit of applying ICA on the innovation process rather than on the original data is that the innovations are usually more independent and more non-gaussian than the original processes [12].

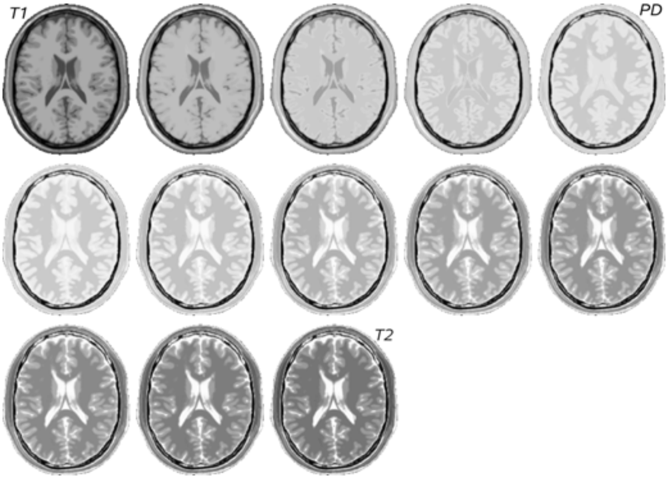


Fig. 1. Simulated brain MR images used in this study

The FastICA algorithm [13] was run one hundred times with different initialisations, to obtain a reliable set of ICs (see [14] for a description of such consistency approaches). From this set, only the first components are selected as SOM inputs.

3.2 Self-Organising Maps

The self-organising maps perform a lattice projection that preserves similarity information from the input space, through competitive and Hebbian learning rules [6]. After training, the spatial locations of the neurons in the lattice are indicative of the intrinsic statistical features contained in the input patterns; the continuous input space is mapped on a discrete set of prototype vectors. Yet, results may vary for different initialisations of the lattice.

To produce quantitative descriptions of data properties, interesting groups of map units, *i.e.*, clusters, must be selected [15]. Clustering map units, instead of the original data has the significant advantage that the set of prototypes can be significantly smaller than the original data set, resulting in a computational cost reduction. In clustering based on local minima, the centroids of the clusters are chosen to be the local minima of the SOM [15]. A map unit is a local minimum if its average distance to its neighbouring map units is larger than any of the corresponding distances to its neighbours. The rest of the map units are then assigned to the cluster of the nearest centroid in the Euclidean sense.

SOM representations of the data may vary for different initialisations of the lattice. In order to find a set of training labels, we exploited this variability. After training multiple randomly initialised SOMs, voxels that grouped consistently together were selected as representing stable properties in the input space. This is followed by a very limited human interaction, to join groups corresponding to identical tissue types, hence creating a set of labelled "super voxels". This improves the performance by reducing the number of clusters used in the classification, without any major change in the overall outcome.

3.3 Discriminative Clustering

DC [7] is used in the final voxel classification. It assumes that data comes in pairs $(\boldsymbol{x}, c)$. During learning, the primary data vectors $\boldsymbol{x} \in R^n$ are paired with auxiliary label data c, *i.e.* the "super-voxels" found with SOM. DC is particularly suited to obtain a good evaluation of degenerative lesions, where there is a continuous transition between healthy and lesioned tissues, like demyelinated white matter in MS. Such is made possible through a distance measure within the clusters. This technique is also different from classification in that the number of clusters is not constrained to be equal to the number of classes c. The combination of these properties – the distance measure and the ability to have more clusters than classes – makes DC perfect for tissue segmentation problems, since different combinations of tissues in a voxel occur, especially on tissue borderlines. Other classification techniques were experimented, for instance support-vector machines ([16]), but their results were far less impressive than the ones obtained with DC. For this reason and due to the limited space avaliable, these results are not presented here.

The label data is used to define relevance in the primary data space [7]. DC partitions this space by interesting variation, with the latter measured by homogeneity of the auxiliary data [17]. Important variations in $\boldsymbol{x}$ are revealed by the conditional density. The goal is then to partition the primary data space into clusters that are local and homogeneous in terms of their auxiliary data [7]. Locality is enforced by defining the clusters as Voronoi regions in the primary data space: $\boldsymbol{x} \in V_j$, if $\|\boldsymbol{x} - \boldsymbol{m}_j\| \leq \|\boldsymbol{x} - \boldsymbol{m}_k\|$ for all k. The Voronoi regions are uniquely determined by the parameters $\{\boldsymbol{m}_j\}$. Homogeneity is enforced by assigning a distributional prototype $\psi_{ji} = p(c_i|\boldsymbol{x}, \boldsymbol{x} \in V_j)$ to each Voronoi region j, and by searching for different sets of partitions, capable of predicting auxiliary data with the prototypes.

The resulting model is a piecewise-constant generative model for $p(c|\boldsymbol{x})$, where its logarithmic posterior probability is maximised [17]. The cluster "membership function" can be of a gaussian-like form [7], where the smoothness is controlled by the standard deviation σ. It was set to 0.3.

A problem with pure DC is that the categories may over-fit to apparent dependencies in a small data set. One of the regularisation methods for DC consists in favouring equal distribution of data into the clusters [18]. This allows to reduce the over-fitting and to overcome the "dead unit" problem, common in k-means after bad initialisation. In order to get the best results, the same amount of "super-voxels" is used for every tissue, avoiding biased results. The regularising parameter was also chosen through cross-validation.

The DC classification was done with three (*normal-set*) and four (*lesion-set*) cluster prototypes, corresponding to the number of tissues present in the data, leading to the best visual data representation. If the number of prototypes is too high, the distinction of the different tissues would lead to imperceptible visual results. On the other hand, if the prototype number is low, the clusters would start to join different classes, rendering the classification useless. The initial points for the prototypes of the discriminative clustering were calculated using *k-means.*

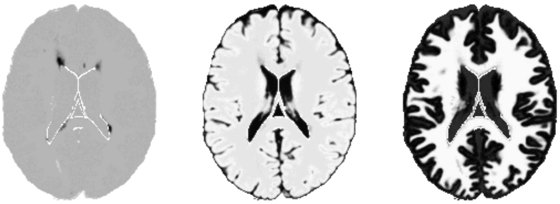


Fig. 2. Independent components obtained from the *lesion-set*. Each image highlights a particular tissue: lesion on the left, CSF in the centre and grey matter on the right.

Table 1. Numerical results of the ”super-voxels“ in the *lesion-set*

Tissue		CSF	White	Gray	MS Lesion
normal	*# of ”super-voxels“*	1134/1438	6464/9369	3841/7213	–
set	*Correct classification*	100%	99.9%	99.5%	–
lesion	*# of ”super-voxels“*	901/2944	4526/9429	2706/6855	102/57
set	*Correct classification*	100%	100%	100%	50%

4 Results

4.1 Image Pre-processing

After masking and registration, ICA on innovations was run in the 13 images of the *lesion-set*, to verify if the method provides good results in these cases. Fig. 2 shows the components found. In the case of the *normal-set*, since only three sequences were avaliable, ICA was not used.

4.2 Super-Voxels

After pre-processing, the “super-voxels” were obtained by sorting the grouped results of multiple SOM runs. In the *lesion-set*, the accuracy of the selection falters only for the MS lesion. This lower reliability may be due to its degenerative nature.

Using only three sequences from the *lesion-set* (PD, T1 and T2), it is impossible to find lesion ”super-voxels“. This fact corroborates the claim that multispectral images are of great importance in lesion detection.

4.3 Tissue Classification

We start by analysing the results in the *normal-set*. The results are measured in terms of the misclassification rate, *i.e.*, the ratio between the correctly classified voxels and the total number of voxels classified. In order to calculate this measure, we used a ”hard“ classification setting where one voxel belongs to the class with more contribution in that voxel, *i.e.* $p(x \in c_k) > p(x \in c_{i,\forall i \neq k})$.

Normal-set: The results for the *normal-set* are shown in Fig. 3 and in Tab. 2. From visual inspection, it is clear that the DC results are consistent and with a good correspondence to the tissues. Examining the numerical results, the method

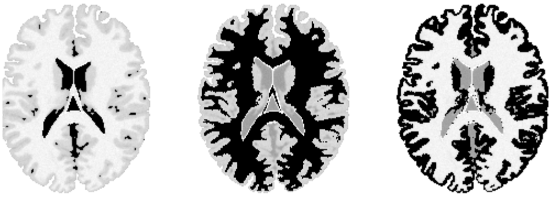


Fig. 3. Classification result for each class in the *normal-set*. CSF, white and grey matter from left to right. The classification is shown overlaying a T2 sequence. Voxels in black correspond to the voxels that have most of their membership in that class.

Table 2. Numerical results of the DC classification. The percentages shown correspond to the amount of voxels correctly classified.

Tissue	CSF	White	Gray	MS Lesion	Misclassification rate
normal-set	99.70%	97.05%	96.42%	–	3.11%
lesion-set	96.61%	98.90%	98.26%	34.38%	2.61%

achieves very good results, with a small percentage of error 3.11%, lower than the best result obtained using the method [8], where data with 5% of noise was used.

Lesion-set: The classification results for the *lesion-set* are presented in Tab. 2 and Fig. 4(a). DC reaches a global misclassification rate of 2.61%. This result is in accordance with the one obtained in the *normal-set*, since there is no noise but there is an extra class (MS lesion). Looking only at the results of the lesion voxels (34.38%), it seems that the method does not perform particularly well for this tissue type. Since PVI for all the tissues was avaliable in the ground-truth, we analysed the causes for such a poor classification rate.

One of the great advantages of DC is the membership classification for each voxel to each class. Since the misclassification rate ignores PVI, taking into account only the class with more contribution in each voxel, we used the root mean square (RMS) error instead [3] $\left(rms\_error_k = \sqrt{\sum_i \left(\widehat{x}_{ik} - x_{ik}\right)^2 / N}\right)$ to evaluate the results, where $\widehat{x}_{ik}$ and x_{ik} are the estimated and true partial volumes of the class k at voxel i, and N is the number of voxels with nonzero partial volume for the class k. We compared the RMS error for each class, when doing a hard comparison, *i.e.* each voxel is assigned only to one class, and when doing a soft comparison, *i.e.* each voxel can have contributions from several classes. This comparison is depicted in Tab. 3. The results obtained with this measure are elucidative about the importance of using a classification method that provides partial volume estimation. The error in the lesion decreased by more than 70%, while also decreasing in the other classes.

Of surmost importance is that the error of the MS lesion came to the same level as all the other tissues. This means that, by outputting membership information for each voxel, DC permits to better evaluate tissue transitions. This corroborates our statement that the DC algorithm is a suitable algorithm for tissue classification and particularly for lesion detection.

Table 3. RMS error in the *lesion-set*

Tissue		CSF	White	Gray	MS lesion
RMS error	hard	0.6198	0.4026	0.5502	1.2776
	soft	0.4119	0.3159	0.3941	0.3790

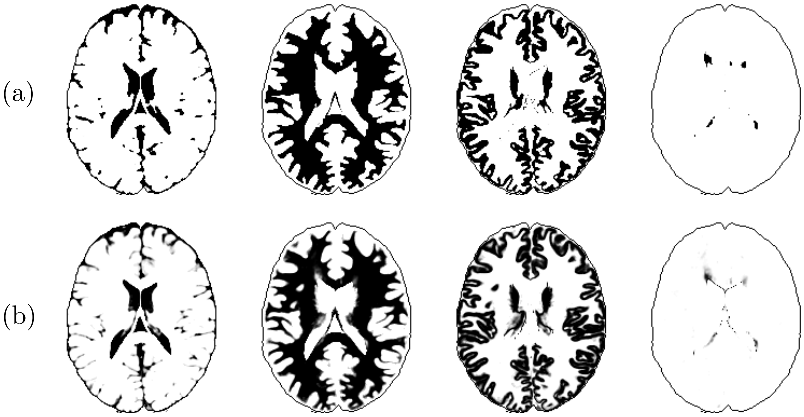


Fig. 4. DC (top) and ground-truth (bottom) partial volume information for each voxel, surrounded by a brain outline. The darkness in each voxel is directly proportional to the percentage of tissue in that voxel (except for the outline). It can be easily seen that all the voxels classified as lesion in DC have a high contribution of lesion, indicating the tendency of the algorithm to show the lesion and it's directions of propagation.

In Fig. 4(b) it is possible to observe the contribution of the lesion to each voxel. It is easily observed that the voxels considered by DC as lesion are voxels that have a large percentage of lesion. This explains the abrupt decrease in the RMS error for the lesion exhibited in Tab. 3 and is a very good indicator since provides information about possible spreading directions of the lesion.

5 Discussion and Conclusion

The semiautomatic tissue segmentation approach uses consistent ICA on innovation as a denoising method. This is followed by creating multiple randomly initialised SOMs, used to estimate a set of labelled data for classification training.

The results of the *normal-set* demonstrate the robustness of the algorithm in correctly classifying different tissues. Also, in the *lesion-set*, all the tissues are well discriminated, and the direction of progress of the disease can be determined.

Another important factor is that human interaction was kept minimal. It was only needed to select the final set of the "super voxels", by analysing all the clusters obtained by SOM and grouping them to the corresponding tissues.

Several assumptions made in this work may be questioned. Yet, the results attained support the use of multi-resolution MRI data and the proposed methodology for the isolation of pathologies. One future step might be to evaluate which sequences are most important for MS lesion detection, which would help in selecting the MR sequences to be used in real world applications.

The soft classification of DC makes it perfectly suited to evaluate different stages of lesions, allowing a better and more refined way to analyse them than typical tissue classification methods. Using the DC results, together with the original and ICA images, experts have a method that allows for better diagnoses and may help dissipating more dubious evaluations. Further research would also address the need for volumetric data, as well as follow-up of pathologies and treatment.

References

1. Hornak, J.: The Basics of MRI (1996), `http://www.cis.rit.edu/htbooks/mri`
2. Atkins, M.S., Mackiewich, B.T.: Fully automatic segmentation of the brain in MRI. IEEE Trans. Medical Imaging 17(1), 98–107 (1998)
3. Choi, H., Haynor, D., Kim, Y.: Partial volume tissue classification of multichannel magnetic resonance images-a mixel model. IEEE Transactions on Medical Imaging 10(3), 395–407 (1991)
4. Karp, E., Gävert, H., Särelä, J., Vigário, R.: Independent Component Analysis Decomposition of Structural MRI. In: Proceeding 2nd IASTED Int. Conf. on Biomedical Engineering (BioMED 2004), Innsbruck, Austria, pp. 83–87 (2004)
5. Hyvärinen, A., Karhunen, J., Oja, E.: Independent Component Analysis. John Wiley & Sons, Inc, Chichester (2001); Simon Haykin (ed.)
6. Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer, Heidelberg (2001); Huang, T.S., Kohonen, T., Schroeder, M.R. (eds.)
7. Kaski, S., Sinkkonen, J., Klami, A.: Discriminative clustering. Neurocomputing 69(1-3), 18–41 (2005)
8. Tohka, J., Krestyannikov, E., Dinov, I.D., Graham, A.M., Shattuck, D.W., Ruotsalainen, U., Toga, A.W.: Genetic algorithms for finite mixture model based voxel classification in neuroimaging. IEEE Trans. Medical Imaging 26(5), 696–711 (2007)
9. Kwan, R.K.S., Evans, A.C., Pike, G.B.: MRI simulation-based evaluation of image-processing and classification methods. IEEE Trans. Medical Imaging 18(11), 1085–1097 (1999)
10. Database, B.S.B.: `http://www.bic.mni.mcgill.ca/brainweb/`
11. Keegan, B.M., Noseworthy, J.H.: Multiple sclerosis. Annual Review of Medicine 53, 285–302 (2002)
12. Hyvärinen, A.: Independent Component Analysis for Time-dependent Stochastic Processes. In: Proc. Int. Conf. on Artificial Neural Networks (ICANN 1998), Svökde, Sweden, pp. 541–546 (1998)
13. FastICA: MATLAB™ package (1998), `http://www.cis.hut.fi/projects/ica/fastica`
14. Ylipaavalniemi, J., Vigário, R.: Analyzing consistency of independent components: An fMRI illustration. NeuroImage 39(1), 169–180 (2008)
15. Vesanto, J., Alhoniemi, E.: Clustering of the Self-Organizing Map. IEEE-NN 11(3), 586 (2002)
16. Karp, E., Vigário, R.: Unsupervised MRI Tissue Classification by Support Vector Machines. In: Tilg, B., Proceeding 2nd IASTED Int. Conf. on Biomedical Engineering (BioMED 2004), Innsbruck, Austria, pp. 88–91 (2/16/2004 - 2/18/2004)
17. Sinkkonen, J.: Learning Metrics and Discriminative Clustering. PhD thesis, Helsinki University of Technology (2003)
18. Kaski, S., Sinkkonen, J., Klami, A.: Regularized discriminative clustering. In: Neural Networks for Signal Processing XIII, pp. 289–298. IEEE, New York (2003)

Time Series Analysis for Long Term Prediction of Human Movement Trajectories

Sven Hellbach[1], Julian P. Eggert[2], Edgar Körner[2], and Horst-Michael Gross[1]

[1] Ilmenau University of Technology, Neuroinformatics and Cognitive Robotics Labs,
POB 10 05 65, 98684 Ilmenau, Germany
sven.hellbach@tu-ilmenau.de
[2] Honda Research Institute Europe GmbH, Carl-Legien-Strasse 30,
63073 Offenbach/Main, Germany
julian.eggert@honda-ri.de

Abstract. This paper's intention is to adapt prediction algorithms well known in the field of time series analysis to problems being faced in the field of mobile robotics and Human-Robot-Interaction (HRI). The idea is to predict movement data by understanding it as time series. The prediction takes place with a black box model, which means that no further knowledge on motion dynamics is used then the past of the trajectory itself. This means, the suggested approaches are able to adapt to different situations. Several state-of-the-art algorithms such as Local Modeling, Cluster Weighted Modeling, Echo State Networks and Autoregressive Models are evaluated and compared. For experiments, real movement trajectories of a human are used. Since mobile robots highly depend on real-time application, computing time is also considered. Experiments show that Echo State Networks and Local Model show impressive results for long term motion prediction.

1 Introduction

For autonomous robots, like SCITOS [1], it is important to predict the motion of people and other robots in their environment, for example to avoid collisions. Hence, further actions can be planned more efficiently. Most approaches in this field focus on optimal navigation strategies [2,3]. This paper suggests to spend more effort into prediction of the motion of the dynamic objects (i. e. in most cases the motion of humans in the scene) instead. Often, only linear approximations or linear combinations are used to solve this problem.

Plenty of algorithms exist for time series analysis and prediction. Their fields of application reach from prediction of economic data to climate and biologic data, such as neural activities [4]. The new approach is the interpretation of movement data as time series to perform a long-term prediction. For this prediction, an assortment of time series analysis algorithms has been implemented and comparatively tested.

For this, it is necessary to know the motion trajectories of the surrounding dynamic objects. For simplification, a tracker is assumed, which is able to provide

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 567–574, 2009.

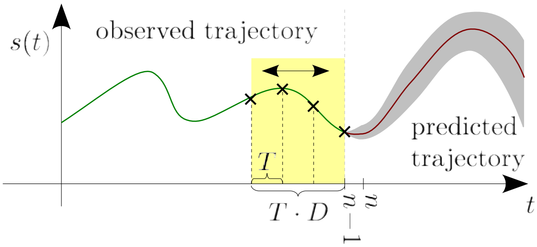


Fig. 1. The observed trajectory (green) is to be predicted (red) for up to 500 time steps (about 8.3 sec. at 60 Hz). This is achieved only by exploiting the past trajectory's characteristics using a window (yellow) of D points equally spaced with interval T.

such trajectories in real-time. A possible tracker to be used is presented in [5]. In this case, the given trajectory of the motion can be interpreted as a time series $\mathcal{T}$ with values $\mathbf{s}_i$ for time steps $i = 0, 1, \ldots, n-1$: $\mathcal{T} = (\mathbf{s}_0, \mathbf{s}_1, \ldots, \mathbf{s}_{n-1})$.

The next section introduces our time series analysis approach to mobile robotics and techniques chosen to be tested. In section 3 the comparing experiments with their conditions and results are presented, while the paper concludes in section 4.

2 Time Series Prediction

The algorithms presented in this paper are intended to be used for motion prediction to enable a more anticipative mobile robot navigation in dynamic environments. Basically, for all presented algorithms the prediction for each future point on the trajectory is done iteratively for up to 500 time steps (this corresponds to about 8.3 sec. of motion with a sampling frequency of 60 Hz) (Fig. 1).

The prediction in general takes place with the so-called black box model which means that no further background information or knowledge about the motion dynamics is used than the past trajectory itself. The aspired prediction is to follow the trajectory's characteristics, only, which can be found in their past. Furthermore, no explicit trajectory models are given, to be able to freely adapt to yet unknown situations.

2.1 Echo State Networks

For prediction of time series, Echo State Networks are often used [6]. They have some specific features which differ from "standard" neural networks: The hidden layer consists of neurons which are randomly connected. When the connectivity is low, this layer provides independent output trajectories. For this reason, the hidden layer is also called "reservoir". Furthermore, there are neurons which are connected to loops in the reservoir, so that past states "echo" in the reservoir. That is the reason, why only the actual time series value is needed as input.

Another characteristic of Echo State Networks is that only the output weights are adapted and learned. All other weights (input, reservoir, feedback) are chosen randomly and stay statically. For training, the net is randomly initialized, and the training time series is used as net input step by step.

This paper suggests to use multiple instances of the network, as a kind of simple stochastic search in the parameter space. The fixed weights are initialized differrently in a random manner. All instances are trained using the same input data. During the training process, the output of each network is compared with the corresponding values of the training trajectory. The network showing the best prediction results for the yet unknown training data is then selected for further application.

2.2 Autoregressive Models

The next type of time series analysis algorithms introduced here are Autoregressive Models (AR). These models assume a linear relation in the time series which means that any time series value can be determined by using a linear combination of p previous values. The coefficients of the linear combination – the AR coefficients – have to be calculated to predict future values. Several Algorithms exist to determine these coefficients, e. g. Wiener Filter [7], Durbin-Levinson [8], and Yule-Walker [8].

2.3 Embedding Space

For applying the approaches in sections 2.4 and 2.5, an embedding in a higher dimensional space is necessary. This embedding can be regarded as a kind of the well known sliding window approach. An observation window with size $T \cdot D$ is put on the trajectory (Fig. 1). Each T-th time step from this window is used to generate this *regular embedding*. So the time series is transformed into a D-dimensional space - the *embedding space*. To each embedding $\mathbf{e}_t = \left(\mathbf{s}_t, \mathbf{s}_{t-T}, \mathbf{s}_{t-2T}, \ldots, \mathbf{s}_{t-(D-1)T}\right)^T$ belongs an output $\mathbf{o}_t$, which stands for the successor $\mathbf{s}_{t+1}$ of the selected window.

The two introduced parameters T and D don't need to be defined by hand. Time series analysis offers techniques to automatically determine these parameters [4]. In our work, we used genetic algorithms to find the best suited embedding dimension.

2.4 Local Modeling

Local Modeling, which is described in [9], is based on the aforementioned regular embedding. The principle idea is a simple nearest neighbor search in the embedding space of the last point in the time series $\mathbf{e}_{n-1}$ for which the prediction needs to be calculated.

In the general case, a polynomial is estimated for prediction describing the relationship between embedding $\mathbf{e}_i$ and output $\mathbf{o}_i$. The nearest neighbors are used to decide the polynomial's coefficients $\mathbf{v}$ applying linear regression. In practice, the polynomial degree g is usually low. Often it is enough to use $g = 0$ (*Local Averaging Model*) or $g = 1$ (*Local Linear Model*).

After determining the coefficients, the prediction is calculated using the same polynomial interpolation. To get good prediction results, it is crucial to choose

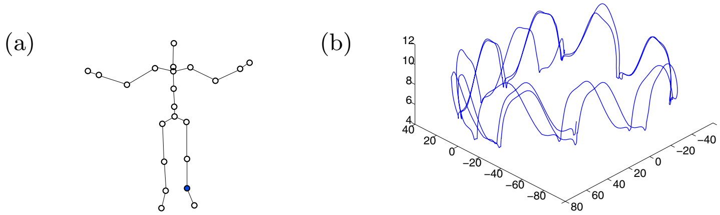


Fig. 2. Example of movement data from the University of Glasgow. Data is available for the body limbs shown in (a) and an exemplary trajectory of the movement of the left ankle while walking in circles (b).

proper parameters, such as the embedding parameters T and D and the number of the nearest neighbors N. Especially with higher polynomial degrees, the algorithm is extremely sensitive to the choice of these parameters. Therefore, an evolutionary algorithm was implemented which often leads to good results as recommended in [9].

2.5 Cluster Weighted Modeling

The Cluster Weighted Modeling, which is described in [9], is also operating in the embedding space. The viewpoint lies not on single embedding points like in the Local Modeling. Now the embedding space is clustered and covered with Gaussians.

An Expectation-Maximization-algorithm (EM-algorithm) can be used to optimize most of the algorithm's parameters. The whole algorithm can be found in detail in [9]. Only the number of clusters and the cluster function remain to be chosen manually. All other parameters are initialized randomly and adapted using the optimization. As cluster function, similar functions like the Local Modeling polynomials, can be used. Since, calculation time strongly depends on the number of clusters, the values of these parameters should not be too high for an online application.

3 Motion Prediction

The algorithms presented in this paper are intended to be used for motion prediction to enable a mobile robot navigating in dynamic environments. To be comparable and reproducible, movement data taken from the University of Glasgow [10] is used. This benchmark data is available as 3D coordinate representation for each limb of a human performing a certain action, e. g. walking (see Fig. 2). Using this data is even more challenging, because several basic motions are combined (i. e. intrinsic movement, e. g of the foot combined with the walking direction). The data set consists of 25 trajectories containing 1,500 up to 2,500 sampled points in Cartesian space.

3.1 Test Conditions

The movement data has a resolution of 60 time steps per second, so that an average prediction horizon of about 500 steps corresponds a prediction of 8.3

seconds into the future at 60 Hz. Present movement prediction techniques are designed to predict an objects position for the next time frame or at least to gap a loss of the object during a only a few frames.

Quality Measures. For comparing the prediction results, some kind of quality measures for comparison are necessary. The used quality measures are based on the normalized mean square error $NMSE$. Hence, the standard mean square error is normalized using the variance σ^2 of the time series.

$$NMSE = \frac{1}{N \cdot \sigma^2} \sum_{i=1}^{N} (\mathbf{s}_i^{pred} - \mathbf{s}_i^{orig})^2 = \frac{MSE}{\sigma^2} \tag{1}$$

Since the trajectories are three-dimensional and dimensions with greater difference suppose to be more important, the highest variance of all dimensions is used as normalization.

Two different kinds of the defined measure are used. The first one, the short term error STE, is responsible for evaluating a short period of the prediction. It uses the first $N = 75$ prediction steps (which means 1.25 sec) with a weighting of $\frac{1}{f}$ for the f-th prediction step. On the other hand, the performance is evaluated using the long term error LTE, which uses all prediction steps with a weighting of $1/\sqrt{f}$, since some of the algorithm show the tendency to drift away

Reference Algorithms. Additional simple reference algorithms were used that should be outperformed clearly to get a useful prediction. The first algorithm is a simple repetition of the last time series value and is called constant algorithm in the following. Also a linear algorithm is used as reference. This algorithm simply does a linear approximation in the last two points in the time series. The result of the better one is used as reference.

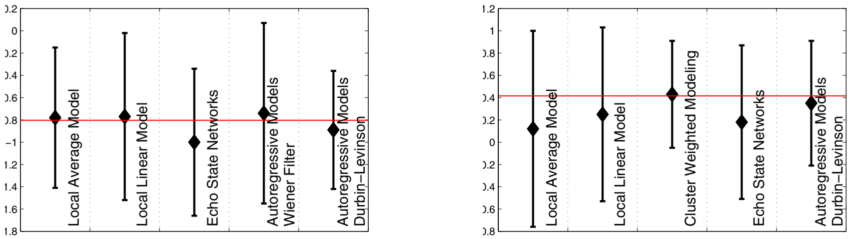
3.2 Experimental Results and Comparison

The following tests show the advantages and disadvantages of the different algorithms presented here. For the application, a number of parameters had to be decided to apply the algorithms. The values presented in the following are chosen after extensive test, which are not discussed here.

Especially for Echo State Networks the choice of the parameters is important. It has shown that the scaling of the weights is essential. The feedback weights $\mathbf{w}_{back}$ must be scaled very low (ca. 10^{-20}) to guarantee stable networks dynamics.

As input for the Wiener Filter the embedding presented in section 2.3 is used instead of only using the last p values. Experiments show that using the embedding leads to better results. For the other Autoregressive Models values around $p = 100$ for AR depth often lead to the best results.

For generating the embedding, the number of histogram bins for calculating the mutual information has to be specified. Proper values are between 15 and 30. In most cases, the smaller value is used to keep the calculation time low. To fasten the whole embedding procedure, not every embedding point is used for



(a) (b)

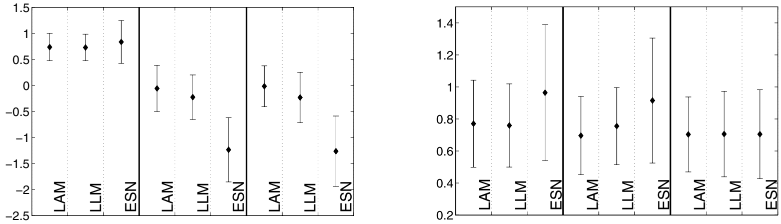
Fig. 3. The graphs shows the STE (a) and LTE (b) plotted for each of the investigated algorithms. The ordinate is scale logarithmically. Hence, lower values mean a better prediction. The error bars represent the standard deviation from the mean. For the STE, all results lie relatively close together while the reference algorithm (red line) can only beaten clearly by the Echo State Networks. Longer predictions show more differences in the results of the algorithms. Also the mean errors are higher than STE, as being expected in longer predictions. The reference is beaten more clearly in general. Local Average Models (LAM) and Echo State Networks show the best results.

the classification in true and false neighbors, but a random selection of around 5-10% of the time series embedding points.

In the prediction of movement data, the Echo State Networks lead to the best results for the STE as it is shown in Fig. 3(a), while for long term prediction Local Models have slightly better results (Fig. 3(b)). The Autoregressive Models perform barely better than the reference. Here the Durbin-Levinson algorithm achieves the best prediction quality. Cluster Weighted Models show the worst performance, and their mean errors stay even behind the simple reference algorithms. The best algorithms still beat the simple references clearly and are able to predict movements for several seconds (about 100 prediction steps) very well.

The choice of the number of neurons in the Echo State Network reservoir, for example, has only a minor effect. In tests the difference in the prediction results of movement data between 25 and 250 neurons were insignificant. It can be presumed that the structure of the movement data does not allow a higher accuracy in the prediction unlike other chaotic time series [6].

The evaluation discussed in the previous paragraphs used a time horizon of 1000 time steps for training. Towards, online application such a long training phase would mean to observe the person for several seconds. Since, this is not possible in most cases, the tests depicted in Fig. 4 are tested with less data. Only 300 time steps of the trajectory are used now. Those 300 points in time are subsampled for the three left most results in Fig. 4(a) and Fig. 4(b), as it would be the case when using a slow tracker. As it can be expected, the prediction quality significantly decreases (compared to the three right most results in 4(a) and Fig. 4(b)). A logical step at this point is to use interpolation to fill the missing gaps. A spline interpolation is used for the test in Fig. 4 to gain 300 time steps of training data again. The results can be compared to the ones using the original trajectory (see the three midway results in 4(a) and Fig. 4(b)).



(a) (b)

Fig. 4. The graphs show the *STE* (a) and *LTE* (b) plotted for the most promising algorithms of the previous tests (Local Average Model (LAM), Local Linear Model (LLM), and Echo State Networks ESN) in a similar fashion as in Fig. 3. Each plot is separated into 3 sections. From left to right, these sections show the results for the test with the subsampled trajectory, the interpolated trajectory, and the comparison with the normal trajectory.

Calculation Time. For any online application, the calculation time plays a big role, since the movement is supposed to be predicted before it continues. Since, only MatLab implementations were tested on time series with lengths around 1,000 till 2,500 time steps, only a first hint can be given here.

Autoregressive Models and Echo State Networks with lower number of neurons show a calculation time of about 3-10 ms per prediction step. This is absolutely complying with online requirements.

Local Models and Cluster Weighted Models need longer calculation times between 50 and 250 ms. In the first case (Local Models), most calculation time is spend on the search for the nearest neigbors in the high number of training data. The Cluster Weighted Models are slow because of a long optimization time (the EM-algorithm).

4 Conclusions and Future Works

The intention of this paper was to connect the well-known field of time series prediction and movement data handling from robotics in a consistent way. Different behaviors from the tested time series analysis algorithms were observed. Generally, it can be said that movement data behaves different than periodical and chaotic time series.

The tested algorithms show very good results in predicting several seconds of the movement data. Echo State Networks and Local Models pointed out to ba a suitable algorithm for movement prediction

Autoregressive Models and again Echo State Networks are able to predict fast enough for an online application without any further adaptation. From the current point of view, Echo State Networks are the ”winning” approach which is able to solve the problem best. Hence, further analysis should have the focus on this approach and on additional improvements.

The other algorithms can be upgraded as well. Local Models can be a good alternative to Echo State Networks if they could be accelerated without loss of quality. Besides this, enhanced versions of the Autoregressive Models such as ARMA or ARIMA Models could be tested. Furthermore, the usage of an irregular embedding is imaginable.

As a next step, an adequate navigation strategy exploiting the prediction results needs to be investigated. One drawback for predicting motion data is the fact that human beings may perform unexpected motion. Since the discussed algorithms rely on the known characteristics, it is possible to use them for detection of such unexpected behavior.

Acknowledgment

Thanks to our students Sören Strauß and Sandra Helsper for doing the evaluation work and contributing good ideas.

References

1. Gross, H.M., Böhme, H.J., Schröter, C., Müller, S., König, A., Martin, C., Merten, M., Bley, A.: Shopbot: Progress in developing an interactive mobile shopping assistant for everyday use. In: Proc. IEEE Internat. Conf. on Systems, Man and Cybernetics, IEEE-SMC (to appear) (2008)
2. Pett, S., Fraichard, T.: Safe navigation of a car-like robot within a dynamic environment. In: Proc. of European Conf. on Mobile Robots, pp. 116–121 (2005)
3. Owen, E., Montano, L.: Motion planning in dynamic environments using the velocity space. In: Proc. of RJS/IEEE IROS, pp. 997–1002 (2005)
4. Abarbanel, H., Parlitz, U.: Nonlinear Analysis of Time Series Data. In: Handbook of Time Series Analysis, pp. 1–37. Wiley-VCH, Chichester (2006)
5. Scheidig, A., Müller, S., Martin, C., Gross, H.M.: Generating person's movement trajectories on a mobile robot. In: Proc. of International Symposium on Robots and Human Interactive Communications (RO-MAN), pp. 747–752 (2006)
6. Jaeger, H., Haas, H.: Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless telecommunication. Science, 78–80 (April 2004)
7. Wiener, N.: Extrapolation, Interpolation, and Smoothing of Stationary Time Series. Wiley, Chichester (1949)
8. Shumway, R.H., Stoffer, D.S.: Time Series Analysis and Its Applications. Springer Texts in Statistics (2000)
9. Engster, D., Parlitz, U.: Local and Cluster Weighted Modeling for Time Serie Prediction. In: Handbook of Time Series Analysis, pp. 38–65. Wiley-VCH, Chichester (2006)
10. http://paco.psy.gla.ac.uk/data_ptd.php

Error Analysis of a Sub-millimeter Real-Time Target Recognition System with a Moving Camera

V. M. M. Vieira[1], G. J. Kane[1], R. Marmulla[1], J. Raszkowsky[2], and G. Eggers[1]

[1] Department of oral and cranio-maxillofacial Surgery, Heidelberg University,
Im Neuenheimer Feld 400,
69120 Heidelberg, Germany
[2] Institute for Process Control and Robotics, Universität Karlsruhe (TH)
Building 40.28, Engler-Bunte-Ring 8
76131 Karlsruhe, Germany
Vitor.Vieira@med.uni-heidelberg.de

Abstract. This paper discloses a method for simple and efficient optical coupling of a robotic arm with a tool with unknown location without exerting forces to the tool. Current solutions involve moving the robot in force-control mode and coupling by means of a manual gripper. This poses the problem with the transfer of unwanted forces to the tool while attempting to secure the design. With the intrinsic solution presented here, the camera is placed on the coupling axis and thence measures the distance and orientation to the target, the user will have the ability to safely guide the robotic arm towards the tool and smoothly couple the tool with the robot's end effector. The mechanical prototype is not here described; this paper emphasizes the image processing, consequent data interpretation and general approach. After the explanation of the technique, its theoretical performance limit was examined and confirmed against the practically achieved performance.

1 Introduction

A number of medical robotic tasks require a robot to connect with a patient; this individual task however is not simple and implies a number of constraints for this function to be completed safely. Primarily, this coupling process needs to minimize any forces applied to the patient. The concept pursued in this paper can be transferred to other application areas outside the medical arena, where similar requirements exist: couple and secure a robot arm with a tool / device, in a semi-determined position, without the exertion of forces on the tool or target object.

Several extrinsic solutions based on optical / electromagnetic navigation systems have been previously developed [1] [2] [3], but the use of extrinsic solutions do not offer enough flexibility. Instead requiring a high level of prior knowledge of the workspace, including the location of tools and robot herein; requiring additional tracking hardware, and clear line of sight (in the optical tracking case); they do not allow an easy user-friendly coupling; and finally manual grippers are usually hard to manipulate, requiring additional information on not only target object location, but its exacting structure and precise orientation. These solutions may allow an automatic

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 575–582, 2009.

coupling solution, but forces are applied on the tool and may be measured to assert a correct coupling of both robot and patient-bounded tool. However, these contact forces are also applied to the patient, increasing therefore the hazard of patient trauma.

Here we present a single camera-based solution that allows both automatic and semi-automatic zero force coupling of a robot and target object. With specific graphical symbols placed on the target tool, the camera can measure the distance and orientation to the target. This allows the user the ability to safely guide the robotic arm towards the patient-bound tool with constantly updated trajectory information, thus allowing a constrained path profile minimizing the risk to the patient and allowing a smooth coupling of the tool at the robot's end effector with the target object. This continuous real-time image processing allows corrections in robotic movement and guidance, particularly when the target is not fixed in space.

The mechanical prototype was presented in [4]. This paper's focus within the general approach is the camera's image processing, and subsequent data and error interpretation.

2 Method Overview

In order to implement this concept, particular camera specifications are required: the selected lens must be able to focus at a close range (5-8cm away) and have an acceptably wide view-angle to capture the whole target area without cropping the markers once coupled; the aperture should be small, to obtain a large enough depth-of-field; the light source should be powerful enough to illuminate the target sufficiently, even with a small aperture. The selected lens was a Cinegon 8mm F1.4[1], the camera a Basler A101f[2]. The lens is not auto-focus and the light source is provided by a built-in LED ring with diffuse filter.

The robot end effector consists of a fixating ring, camera and a Force-Torque Sensor. The Force-Torque Sensor allows the user to maneuver the robot arm as he / she sees fit, thus allowing someone to move the end-effector to a gross start position state. The moment the target and its symbols are visible and recognized, restrictions on possible robot movement are immediately applied to ensure the robot moves only within its planned path profile; the restricted movements are either movement along the trajectory to the target and coupling, or away from the target uncoupling. This is the semi-automatic coupling concept.

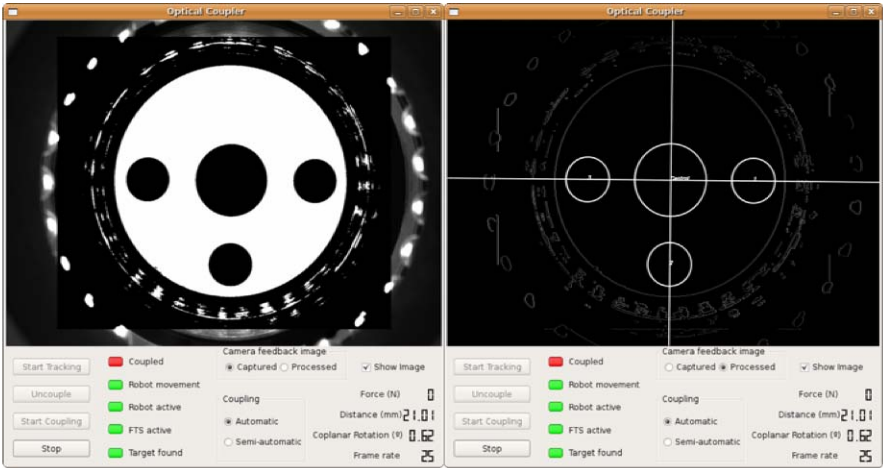
On a flat surface of the target tool a visual pattern consisting of four circles in a known size and configuration is placed. The circles are painted with black non-reflective matt paint, producing a high-contrast optical tracking target for the image processing and are presented in figure 1.

By definition, in close-range photogrammetry the distance from the camera to the object of interest is between one meter to 300 meters [5]. In this application the distance is very close; the markers here are in the range of 3 to 8cm.

The reduction of a three-dimensional object to a two-dimensional image implies a loss of information. Object areas which are not visible in the image cannot be

[1] www.schneiderkreuznach.com

[2] www.baslerweb.com



🟥 Coupled
🟩 Robot movement
🟩 Robot active
🟩 FTS active
🟩 Target found

Camera feedback image
● Captured ○ Processed
☑ Show image

Coupling
● Automatic
○ Semi-automatic

Force (N) 0
Distance (mm) 21.01
Coplanar Rotation (º) 0.62
Frame rate 25

🟥 Coupled
🟩 Robot movement
🟩 Robot active
🟩 FTS active
🟩 Target found

Camera feedback image
○ Captured ● Processed
☑ Show image

Coupling
● Automatic
○ Semi-automatic

Force (N) 0
Distance (mm) 21.01
Coplanar Rotation (º) 0.62
Frame rate 25

Fig. 1. Left: Target detected and applied filter; Right: Target identified and processed

reconstructed from it. This not only includes hidden parts of the tracking target but also regions which can not be recognized due to lack of contrast or limiting size. Circular flat objects in 3D are viewed as ellipses on a 2D image plane; however, their 3D positions can be computed from their projections [6].

2.1 Calibration and Undistortion

A real and practical photogrammetric camera will differ from the traditional pinhole camera model, as presented in figure 2. The necessity of using a relatively complex objective lens, a camera box with an aperture which is not pinhole and the fact that the target will not be parallel to the image plane gives rise to departures from the ideal linear image geometry.

$$m = \frac{h}{c} = \frac{X}{x\,'} \quad . \tag{1}$$

In fact, practical observations note that with the change of depth (h) the scale factor (m) is not constant. For this reason several calibration and undistortion algorithms have been developed and are now common practice [7] [8] [9]. However instead of undistorting these images, in accordance with common practice, the followed

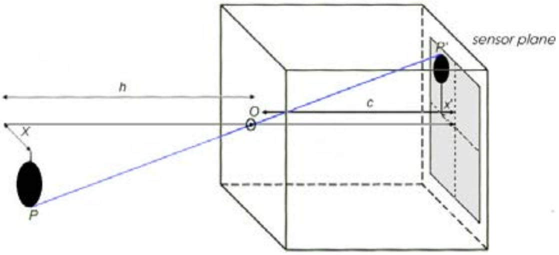


Fig. 2. Pinhole camera model

approach uses the information contained within the distorted image, and expected non-linearity of the circles area to calculate the depth / distance to the target.

The robot was moved in small steps while keeping the image plane parallel to the target plane, several points at different distances from the target where collected and a trend line was calculated using the circle's area in relationship to the distance. To determine the effective distance in mm between the image plane and the target a Platinum FaroArm[3] was used.

With the circle's area and previous knowledge of the target's distance we calculate a polynomial function of second degree, as shown in figure 3.

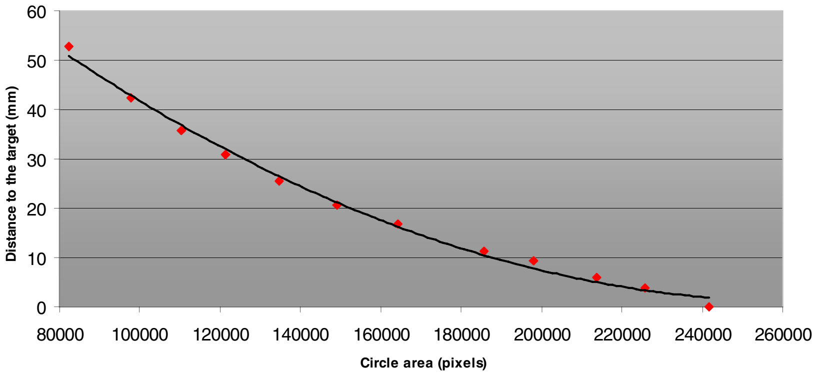


Fig. 3. Relationship between circle area and distance in millimeters

3 Results and Error Analysis

The concept was programmed and tested using OpenCV's[4] Computer Vision libraries, supported by Willow Garage (Menlo Park, USA), which offers an ellipse fitting algorithm based on least squares. Although it is not the most accurate algorithm it is very fast and allows real-time processing [10]. The ellipse fitting error was calculated and used to differentiate the detected contours. Only shapes that have a fitting error under a certain threshold are considered part of the markers.

In average the detected ellipses had a fitting error of 0.166 pixels, where in 95% of the acquired samples the error lies lower than 0.170 pixels. The relationship between the circles size in pixels and mm, is given by a simple function that refers to the known geometry of the object and its perceived size.

$$relationship\ mm\ px = \frac{actual\ \ perimeter\ (mm)}{percieved\ \ perimeter\ (px)} \tag{2}$$

A fitting error of 0.166 pixels corresponds in our case to 0.0026mm in the coupled position. This intermediate result is satisfactory given that the used robot, a Stäubli RX90CR [11], has a repeatability error of 0.02mm, almost ten times larger.

[3] www.faro.com

[4] http://opencv.willowgarage.com/

At points further away from the image plane the mm to px relationship is increasingly larger, growing therefore the theoretical error achieved with this method. However, this coplanar error does not grow out of proportion in smaller tilting angles. In larger angles the highest measurable error was 0.00325mm, which is still very acceptable to our application.

Over 200 collected points in space were collected and calculated to determine this error. The graphical representation is done in figure 4, where these positions are characterized by individual arrows representing the normal of the image plane (camera) and how it perceives the target. The color of the arrow represents the amount of error in that position with that tilt angle.

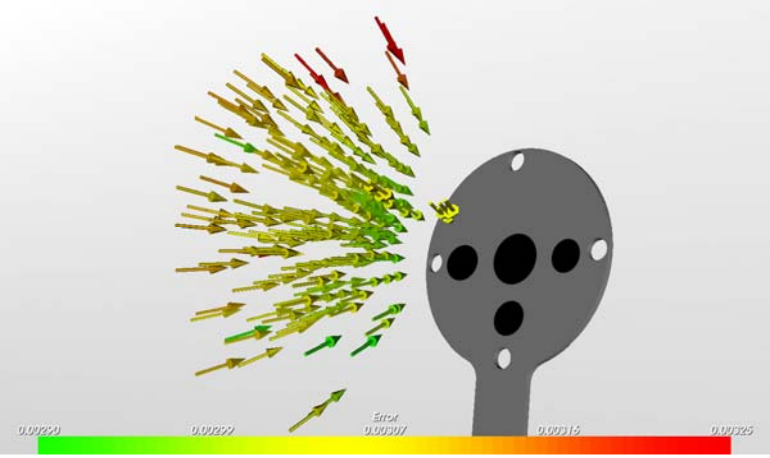


Fig. 4. X & Y position error growth with the camera distance and orientation

A logical conclusion is that the bigger the circles are perceived in the image plane, the lesser the error. That includes skewed circles (ellipses) in disadvantageous positions, which lower the area and amount of data points in the edge of the detected ellipse.

Method used to calculate the depth (distance to the target) is slightly different from the usual methods; the basis for the depth calculation is the use of the previously acquired relationship between the circle's area and its actual distance to the target. Figure 3 illustrates the acquired data and adapted trend line for the larger central circle. Similar data interpretation was done for the 3 smaller circles, obtaining therefore the distance at which each circle is from the image plane. This information is then stored and used to minimize the difference between the expected (target) result and the current image frame.

The image processing calculated values were compared against measured values, obtained using a FaroArm measuring tool, which has a known inaccuracy of 0.013mm [12].

The analysis of these errors has shown that the coupling axis contains the lower errors, in the range of 0.1mm, where as the points further away from this path and tilted regarding the target plane can go up to 3.5mm, confirming that this method is valid and functional.

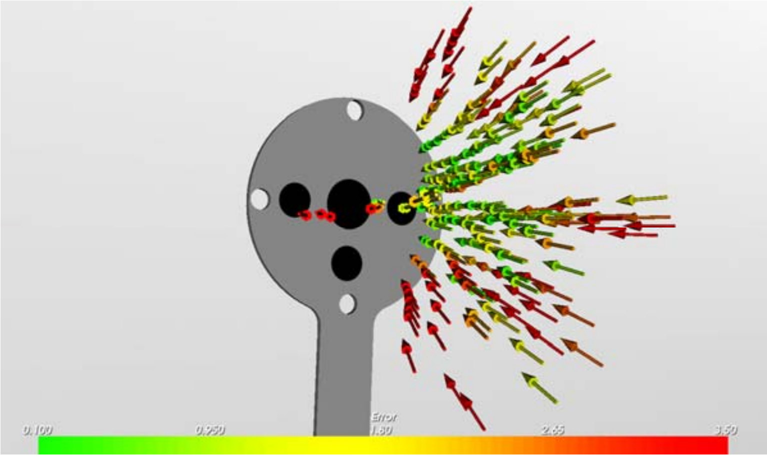


Fig. 5. Depth error growth with the camera distance and orientation

The coplanar angle was also analyzed and shares the same characteristics as the measured inaccuracies of the depth and x/y values. The central axis has the lowest error, 0.1°, while the highest error of 1° can be found in extreme positions, as shown in figure 6.

The low errors in the x/y values and depth values, compared with the slightly higher inaccuracies in the angle value, then supports the chosen approach of the system, already shown before, that the angular error is removed first, in order to achieve higher coupling accuracies overall.

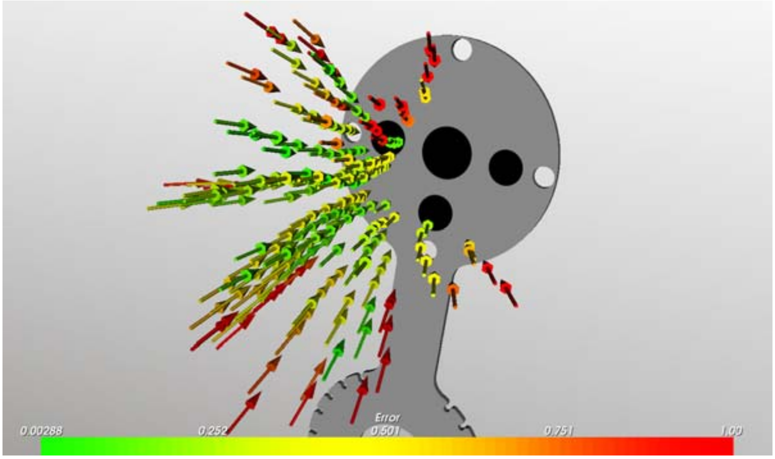


Fig. 6. Angle error growth with the camera distance and orientation

Concluding, the best approach path for the coupling of the tool and robot arm is initially correcting the coplanar angle while driving the arm towards the coupling axis, and then moving forwards towards the target. This path has the lesser error in all three analyzed cases.

4 Conclusion and Discussion

This prototype and required image processing was first engineered to assist in the medical area with the coupling of a robotic arm to a human patient-bounded tool, in a way that sensitive areas do not receive undesired forces from the robot. The concept pursued in this paper can be transferred to other application areas outside the medical arena, where similar requirements exist: couple and secure a robot arm with a tool / device, in a semi-determined position, without the exertion of forces on the tool or target object. It eases the human-robot interaction by enabling the user to maneuver the robot arm as he / she sees fit, thus allowing someone to move the end-effector to a gross start position state. The designed robot end effector simply contains a camera, lens, light source and a force-torque sensor. On a flat surface of the target tool a visual pattern consisting of four circles in a known size and configuration is placed. The circles are painted with black non-reflective matt paint, producing a high-contrast optical tracking target for the image processing.

The approach presented in this article was able to calculate the distance correctly and guide the robot arm safely to the couple position without a requirement to calibrate the camera by conventional methods or undistort the acquired image frame.

The error analysis has shown that the Least Squares algorithm available in OpenCV, achieved an error of nearly two tenths of a pixel, corresponding to 0.0026mm in the coupled position. The depth error was considerably larger, from 0.1mm to 3.5mm (depending on the distance from target) where as the coplanar angle error was in turn between 0.1° to 1° within the visible area of the target.

The error analysis concluded as well that the optimal approach path which minimizes all these three errors is the coupling axis with the image plane parallel to the target plane, and hence this action is completed by the robot first.

The real time trajectory optimization through a single camera can reliably track a 2D marker pattern with enough accuracy to be able to conduct zero force coupling onto a patient bound tool, with no prior knowledge of the robots environment, only the target tool.

Further improvements to this concept include the miniaturization of the target to a more easily handled tool. The use of a smaller target will potentially increase position errors; however, this can be compensated with a higher zoom lens where the targeted circles would be proportionally increased in the image plane. It is acknowledged that complications can arise from this approach requiring the camera and lens to be placed further away from the target, leading to more restrictions on the end effector design.

Secondly, in order to increase the accuracy and stability of the measurements a fifth or sixth circle could be added. Using concentric circles, as suggested by Estaña [13], can as well improve the results. Specifically to increase the accuracy of the depth measurement one could refer to neural networks as another possible path optimization approach.

Acknowledgments. This research work was funded by the Marie-Curie Actions - Early Stage Training, in the framework program number six with the CompuSurge project. A special acknowledgment should be made to the team members of MeGI, IPR Karlsruhe for their support and discussions, without which would not be possible to present this work.

References

1. Burgner, J., Zhang, Y., Eggers, G., Raczkowsky, J., Mühling, J., Wörn, H.: Semiautomatisches Ankopplungsverfahren für einen Assistenzroboter zur Repositionierung osteotomierter Knochenstücke bei orthognath-chirurgischen Eingriffen. In: Curac 2007, Pro BUSINESS GmbH, Karlsruhe, vol. 6, pp. 275–278 (2007)
2. Stolka, P.J., Henrich, D.: Using Maps from Local Sensors for Volume-Removing Tools. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Diego (2007)
3. Furness, D.: System and method for precisely positioning a robotic tool. United States of America (1992)
4. Vieira, V.M.M., Eggers, G., Ortmaier, T., Kane, G.J., Marmulla, R.: Method for optically-controlled semi-automatic coupling of a robot arm with a surgical tool while maintaining sterile conditions. In: Curac 2008, Leipzig, pp. 127–130 (2008)
5. Luhmann, T., Robson, S., Kyle, S., Harley, I.: Close Range Photogrammetry - Principles, Methods and Applications. Whittles Publishing (2006)
6. Kanatani, K.: Geometric computation for machine vision, vol. 37. Oxford University Press, Inc., Oxford (1993)
7. Fiala, M., Shu, C.: Fully Automatic Camera Calibration Using Self-Identifying Calibration Targets. Institute for Information Technology, National Research Council of Canada, Ontario, 26 (2005)
8. Heikkilä, J.: Geometric camera calibration using circular points. IEEE Transactions on pattern analysis and machine intelligence 22, 1066–1077 (2000)
9. Tsai, R.Y.: A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-shelf TV Cameras and Lenses. IEEE Transactions on robotics and automation, RA 3, 323–345 (1987)
10. Gander, W., Golub, G.H., Strebel, R.: Least-Squares Fitting of Circles and Ellipses. Institut für Wissenschaftliches Rechnen, Eidgenössische Technische Hochschule, Zürich, Switzerland, Zürich (1994)
11. Stäubli RX90 specifications (October 08, 2008), `http://www.servosystems.com/staubli_rx90.htm`
12. Advancing measurement - with the complete wireless freedom of a FaroArm (October 08, 2008), `http://measuring-arms.faro.com/distri/start/`
13. Estaña, R., Wörn, H.: Moiré-Based Positioning System for Micro Robots. In: SPIE's Int. Conference on Optical Measurement Systems for industrial Inspection III. Universität Karlsruhe, Deutschland (2003)

Automatic Plaque Boundary Extraction in Intravascular Ultrasound Image by Fuzzy Inference with Adaptively Allocated Membership Functions

Eiji Uchino[1], Noriaki Suetake[1], Takanori Koga[1], Shohei Ichiyama[1],
Genta Hashimoto[2], Takafumi Hiro[2], and Masunori Matsuzaki[2]

[1] Graduate School of Science and Engineering, Yamaguchi University,
1677-1 Yoshida, Yamaguchi 753-8512, Japan
uchino@yamaguchi-u.ac.jp

[2] Graduate School of Medicine, Yamaguchi University,
1-1-1 Minami-kogushi, Ube 755-8505, Japan

Abstract. This paper describes an automatic plaque boundary extraction in the intravascular ultrasound image by a fuzzy inference. In the proposed method, the membership functions in the antecedent parts of the fuzzy rules are adaptively allocated by using the information of the seed points given by a medical doctor. The present method not only improved the accuracy of plaque boundary extraction but also reduced the workload of medical doctors.

1 Introduction

Intravascular ultrasound (IVUS) method is one of the tomographic imaging techniques. The IVUS method, which provides a real-time cross-sectional image of a coronary artery in vivo, is employed for a visualization of an atherosclerotic plaque for a diagnosis of the acute coronary syndromes (ACS) [1]. In the diagnosis of ACS, a precise boundary extraction of plaque is strongly required.

Usually, medical doctors trace the boundary by hand and evaluate the area of plaque for a lot of images. The extraction of plaque boundary is a hard and time-consuming work for a medical doctor. Furthermore, the extraction work is very troublesome because IVUS image is so grainy due to the blood speckle noise. Hence, an automatic precise boundary extraction of plaque is strongly desired.

In the representative conventional boundary extraction method, the seed points given by a medical doctor are interpolated by a spline function [2]. This method can reduce slightly the workload of a medical doctor. However, the accuracy of the interpolation is considerably affected by the number of the seed points and/or a distance between those points.

On the other hand, the methods which are based on automatic search algorithms, e.g., an active contour (snake) model, genetic algorithm, and so on, have been proposed so far [3]. However, those methods involve some iterative processes and take plenty of computing time. To meet with a practical demand in clinic, a quick and automatic boundary extraction by a small number of seed points is definitely necessary.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 583–590, 2009.

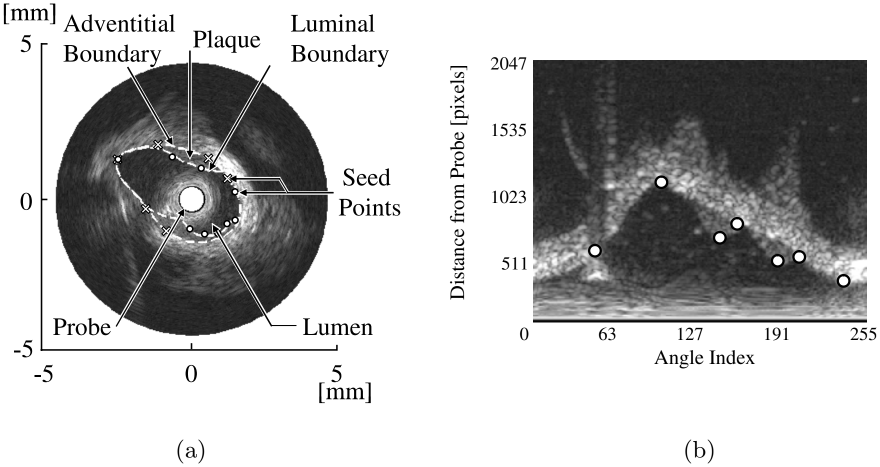


Fig. 1. IVUS B-mode image. (a) The dotted lines show luminal boundary (LB) and adventitial boundary (AB) to be extracted. (b) Transformed B-mode image into the Cartesian coordinates.

The authors have proposed a plaque boundary extraction method [4] by employing a statistical discriminant measure (being called separability [5]) and Takagi-Sugeno (T-S) fuzzy model [6]. The method realizes a fast, precise and robust extraction of the boundary of a plaque. Furthermore, the method does not include any time-consuming iterative processes as in the conventional methods.

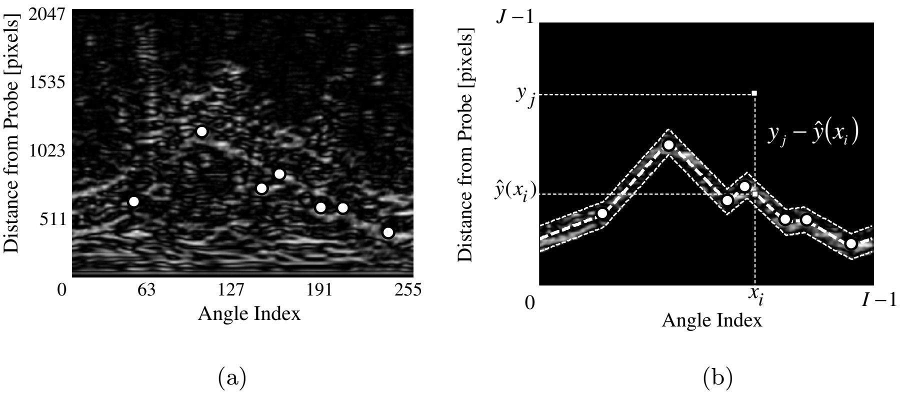
Our previous method [4] has however the following drawbacks; membership functions (MSFs) are allocated uniformly at even intervals, which may cause a bad extraction accuracy; the number of MSFs is decided in a trial-and-error fashion for each IVUS image, which is troublesome for a lot of IVUS images.

In this study, we propose an automatic allocation method of MSFs. In the present method, the number, the widths and the positions of MSFs are decided adaptively according to the complexity of a plaque boundary. The validity and the effectiveness of the proposed method have been verified by applying it to the practical problem of plaque boundary extraction in clinic.

2 Proposed Plaque Boundary Extraction Method in IVUS Image

2.1 Plaque Boundary Extraction by Fuzzy Inference

The image shown in Fig. 1(a) is called "B-mode image." This is a cross-sectional image of coronary artery obtained by IVUS method. The following two boundaries are extracted. One is a luminal boundary (LB) between the lumen and the plaque, and the other is an adventitial boundary (AB) between the plaque and the vascular wall as shown in Fig. 1(a).



(a) (b)

Fig. 2. (a) Discriminant image of the transformed image by using separability. (b) A primitive boundary and a specified search area for the objective boundary.

In this paper, the fuzzy-inference-based boundary extraction method proposed in [4] is improved. In that method, a plaque boundary is approximated by piecewise polynomials inferred by Takagi-Sugeno (T-S) fuzzy model. The boundary extraction procedure is summarized as follows:

1. Seed points are marked on the B-mode image by a medical doctor. The image is then transformed into the Cartesian coordinates as shown in Fig. 1(b).
2. A statistical discriminant measure of separability of the image [5] is calculated. The separability takes a large value around the regional-edge of the image. The brightness of each pixel in the discriminant image is a value of separability for that pixel. Fig. 2(a) shows a discriminant image of Fig. 1(b). That is, a chain of white light pixels can then be a candidate of a boundary.
3. The seed points are first linearly interpolated on the Cartesian coordinates to obtain a primitive boundary (the bold dotted line shown in Fig. 2 (b)). The true boundary is then searched by starting from this primitive boundary. The search area is shown in Fig. 2(b) by the thin dotted lines.
4. The objective true boundary is inferred by T-S fuzzy model, which is piecewise approximated by the polynomials in the Cartesian coordinates by a series of the following fuzzy if-then rules:

   $$\text{IF } x_i \text{ is } A_\ell \text{ THEN } f_\ell(x_i) = a_\ell x_i + b_\ell,$$

   where A_ℓ is a fuzzy set with membership function μ_ℓ shown in Fig. 3. x_i corresponds to the angle index, and $f_\ell(x_i)$ is a linear function. The ℓ-th rule thus stands for the piecewise approximation of the boundary by a linear function in the interval $[t_{\ell-1}, t_{\ell+1}]$. The objective boundary $\hat{y}(x_i)$ is inferred by:

   $$\hat{y}(x_i) = \mu_\ell(x_i) f_\ell(x_i) + \mu_{\ell+1}(x_i) f_{\ell+1}(x_i). \tag{1}$$

   The optimum coefficients a_ℓ^* and b_ℓ^* of the ℓ-th fuzzy if-then rule are determined with use of the weighted least square method (WLSM) so as to minimize the following weighted error criterion:

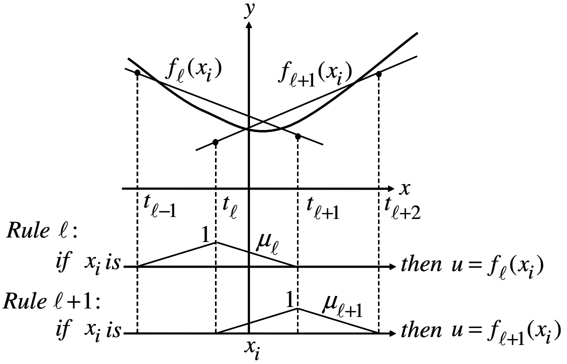


Fig. 3. Membership functions allocated to each $f_\ell(x_i)$, and the fuzzy rules for the interval of x_i.

$$E = \sum_{j=0}^{J-1} \sum_{i=0}^{I-1} \eta_{ij}^2 \{y_j - \hat{y}(x_i)\}^2, \tag{2}$$

where η_{ij} is a separability of pixel (i, j) [4,5]. In this method η_{ij} only inside the boundary search area, being enclosed by the thin dotted lines in Fig. 2(b), are used as the weights of WLSM.

In the next section, the adaptive allocation of the above MSFs are discussed.

2.2 Adaptive Allocation of Membership Functions

Let θ_n $(n = 1, \ldots, N)$ be a specified angle at the n-th seed point as shown in Fig. 4. If all θ_n are obtuse angles, four complementary triangular MSFs are uniformly allocated over the interval of concern. A plaque boundary is then extracted by the T-S fuzzy model. In that case, a smooth plaque boundary is depicted along the obtuse seed points with a small number of MSFs.

In other cases, i.e., acute and obtuse angles exist in the interval of concern, the following procedures are applied for the seed points with acute angles. The seed points with obtuse angles are left unprocessed.

1. Let p_n be an angle index of the n-th seed point, and $d_n^{(R)}$ and $d_n^{(L)}$ be the length between p_n and p_{n+1}, and between p_{n-1} and p_n, respectively. Note that the right-hand side edge and the left-hand side edge of the plaque boundary in Fig. 4 are linked.
2. Two complementary and triangular MSFs are allocated on either side of each seed point. Let $z_n^{(R)}$ and $z_n^{(L)}$ be the tentative peak positions of the MSFs on the right-hand side and the left-hand side of p_n, which are given as:

$$\begin{cases} z_n^{(R)} = p_n + (1 - \rho_n)[d_n^{(R)}/\{2 + \beta(d_n^{(R)})\}] \\ z_n^{(L)} = p_n - (1 - \rho_n)[d_n^{(L)}/\{2 + \beta(d_n^{(L)})\}], \end{cases} \tag{3}$$

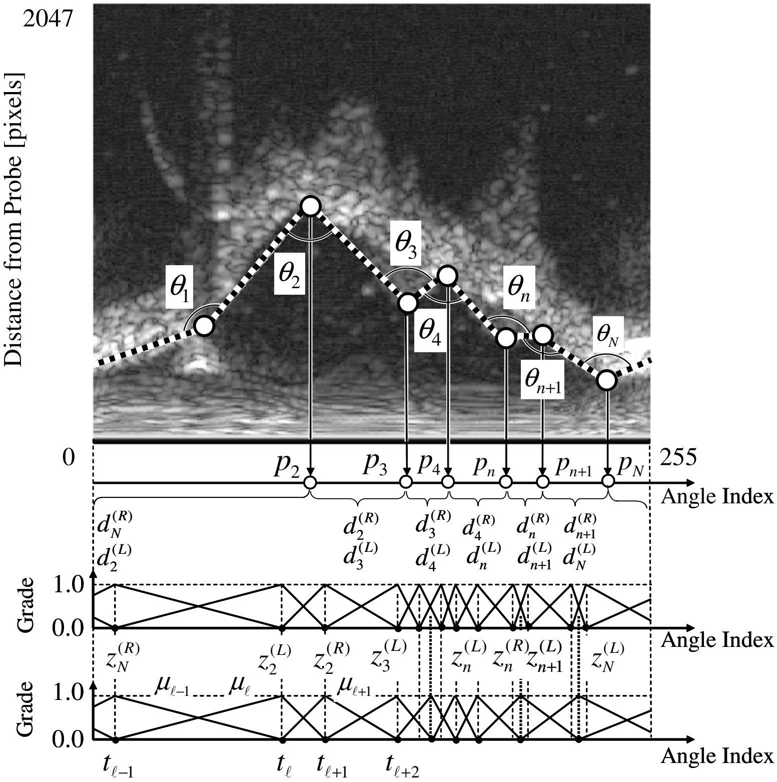


Fig. 4. Procedure of adaptive allocation of membership functions. Some θ_i angles look like to be obtuse, but they are acute. This is because of the length and width ratio of the Cartesian coordinates of the figure.

where ρ_n is a sharpness of the angle θ_n defined by:

$$\rho_n = \begin{cases} 1 - \theta_n/T & for \quad \theta_n < T \\ 0 & for \quad \theta_n \geq T. \end{cases} \tag{4}$$

T is a constant value. The function $\beta(\cdot)$ compensates the peak position of $d_n^{(\cdot)}$ when extremely large, and is defined by:

$$\beta(d_n) = 1/[1 + \exp\{-a(d_n - c)\}], \tag{5}$$

where a and c are parameters to fix the shape of $\beta(\cdot)$. Eq. (3) specifies that when θ_n is small, the MSFs are allocated close to the n-th seed point. This means that the MSFs around p_n are allocated densely, because when θ_n is small, the plaque boundary is complex. The MSFs' allocation map at the upper row in Fig. 4 is thus obtained.

3. The tentatively allocated MSFs above are merged if the following three conditions are satisfied at the same time; (1) the seed point position p_n and the

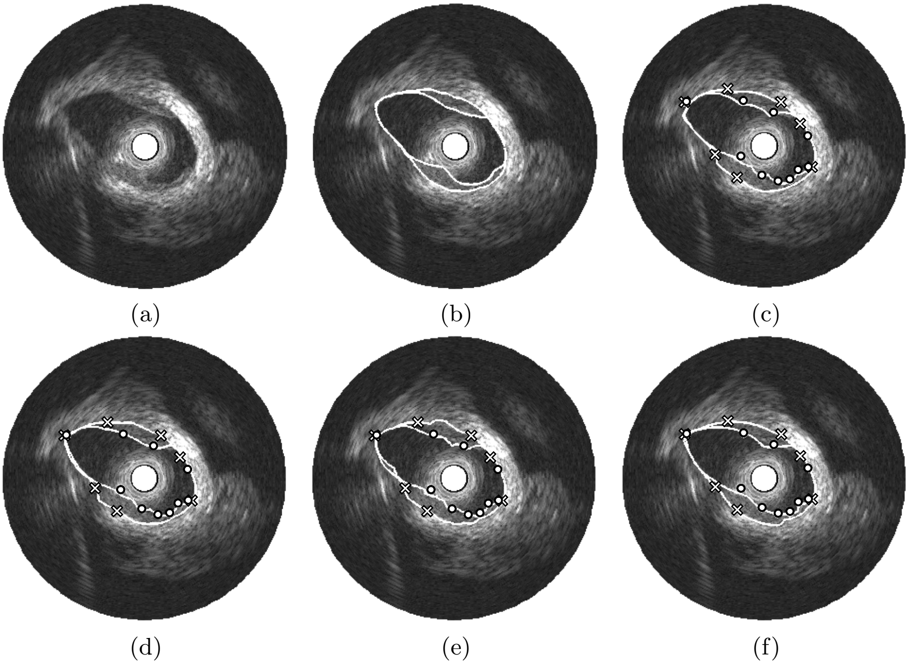


Fig. 5. Boundary extraction results. × : Given seed points for AB (Adventitial Boundary). ○: Given seed points for LB (Luminal Boundary). (a) B-mode image. (b) An objective true boundary to be extracted. (c) Extracted boundary (using 10 fixed MSFs uniformly allocated). (d) Extracted boundary (using 15 fixed MSFs uniformly allocated). (e) Extracted Boundary (using 20 fixed MSFs uniformly allocated). (f) Extracted boundary by the proposed method (using 6 MSFs for AB, and 9 MSFs for LB, adaptively allocated).

peak position of its right-hand side MSF $z_n^{(R)}$ are apart enough each other; (2) the seed point position p_{n+1} and the peak position of its left-hand side MSF $z_{n+1}^{(L)}$ are apart enough each other; (3) the positions of the peaks $z_n^{(R)}$ and $z_{n+1}^{(L)}$ are close each other.

We now define the following criterion to reflect the above three merging conditions:

$$f_1 = 2\|p_n - z_n^{(R)}\|/d_n^{(R)}. \tag{6}$$

$$f_2 = 2\|p_{n+1} - z_{n+1}^{(L)}\|/d_n^{(R)}. \tag{7}$$

$$f_3 = \max(1 - 2\|z_n^{(R)} - z_{n+1}^{(L)}\|/d_n^{(R)}, 0). \tag{8}$$

$$S_{merge} = f_1 \cdot f_2 \cdot f_3. \tag{9}$$

If S_{merge} is greater than 0.5, the peaks $z_n^{(R)}$ and $z_{n+1}^{(L)}$ are merged. An example of merging is shown in the MSFs' allocation map at the bottom row of Fig. 4.

Table 1. Root-mean-square errors (RMSEs) of extraction results

(μm)

	Image 1	Image 2		Image 3		Image 4		Image 5	
	AB	LB	AB	LB	AB	LB	AB	LB	AB
Proposed Method	**41.3**	**33.2**	**27.3**	44.5	**46.4**	**30.6**	**34.5**	**72.7**	**29.4**
10 Fixed MSFs	92.4	41.7	31.6	49.3	47.8	48.3	44.3	95.6	48.6
15 Fixed MSFs	97.5	48.4	37.1	**44.3**	50.3	37.6	40.4	83.1	51.7
20 Fixed MSFs	99.5	46.8	35.9	48.0	59.1	31.6	37.8	91.8	58.5

Table 2. Smoothness evaluation of the extracted boundaries

	Image 1	Image 2		Image 3		Image 4		Image 5	
	AB	LB	AB	LB	AB	LB	AB	LB	AB
Proposed Method	**0.15**	**0.02**	**0.07**	0.95	**0.26**	0.85	**0.64**	**1.09**	**0.72**
10 Fixed MSFs	0.30	0.24	0.14	**0.68**	0.41	**0.79**	0.65	1.31	0.83
15 Fixed MSFs	0.48	0.40	0.48	0.94	0.70	0.95	0.91	1.37	0.87
20 Fixed MSFs	1.14	0.45	0.71	1.39	1.51	1.28	1.23	2.16	1.36

3 Experimental Results

The proposed method is applied to five IVUS B-mode images (Image 1 through Image 5). Extraction results are compared to those by the conventional method with fixed MSFs allocated at even intervals. In the experiments, the parameters in Eqs. (4) and (5) are empirically assigned to be T=45, a=0.05 and c=150.

The parameter T must be tuned by considering the figures of the boundaries of the objective IVUS images. In this experiment, $T = 45$ is employed. On the other hand, the parameters a and c in Eq. (5) are not so sensitive for the extraction of plaque boundaries. This is because a function $\beta(\cdot)$ is used only for a compensation of a peak position of the MFS.

Fig. 5 shows the boundary extraction results for Image 5, one of the five IVUS images employed for the experiments. The results show that the proposed method works very well compared to the conventional method with fixed MSFs uniformly allocated.

The root mean square errors (RMSEs) between the objective boundary $y^*(x_i)$ and the extracted one $\hat{y}(x_i)$ are shown in Table 1. The objective boundary is calculated by a parametric spline-interpolation using a lot of seed points given. It is observed that the proposed method gives better extraction results than the conventional methods with fixed MSFs uniformly allocated.

Further, in order to evaluate the smoothness of the extracted boundaries, the mean of the absolute values of the second order derivatives of the extracted boundary is calculated. The results are shown in Table 2. The superiority of the proposed method is also seen.

Table 3 shows the RMSEs of boundary extraction results in the case of using the same number of MSFs in comparison with the method with fixed MSFs

Table 3. Comparison of RMSEs of boundary extraction results under the same number of MSFs

(μm)

	Image 1	Image 2		Image 3		Image 4		Image 5	
	AB	LB	AB	LB	AB	LB	AB	LB	AB
Number of MSFs	4	4	3	10	4	12	9	9	6
Proposed Method	**41.3**	**33.2**	**27.3**	**44.5**	**46.4**	**30.6**	**34.5**	**72.7**	**29.4**
Fixed MSFs	61.6	**33.2**	43.3	49.3	68.3	38.0	41.9	89.9	78.8

and the method with MSFs adaptively allocated (the proposed method). This implies that the position of each MSF is important for an accurate boundary extraction.

With those results the effectiveness of the proposed method has been verified.

4 Conclusion

We have proposed an automatic plaque boundary extraction in IVUS image by a fuzzy inference. In the present method, the membership functions are adaptively allocated by using the information of the seed points. The present method has shown a better extraction performance than the conventional method in terms of the accuracy and the reduction of the workload of medical doctors.

This work was supported in part by “Knowledge Cluster Initiative,” for 2004-2009, funded by Ministry of Education, Culture, Sports, Science and Technology of Japan.

References

1. Nicholls, S.J., et al.: Intravascular ultrasound in cardiovascular medicine. Circulation 114, 54–59 (2006)
2. Ferguson, J.: Multivariable curve interpolation. J. of the Association for Computing Machinery 11, 221–228 (1967)
3. Klingensmith, J.D., et al.: Assessment of coronary compensatory enlargement by three-dimensional intravascular ultrasound. Int. J. Cardiac Imaging 16, 87–98 (2000)
4. Kubota, R., et al.: Fuzzy rule-based boundary extraction of plaque in intravascular ultrasound image. In: Proc. of the 2008 IAENG Int. Conf. on Imaging Eng., pp. 597–600 (2008)
5. Fukui, K.: Edge extraction method based on separability of image features. IEICE Trans. on Inf. and Syst. E78–D, 1533–1538 (1995)
6. Takagi, T., et al.: Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. on Syst., Man, and Cybern. 15, 116–132 (1985)

Gabor Neural Network Based Facial Expression Recognition for Assistive Speech Expression

Lau Bee Theng

Swinburne University of Technology Sarawak Campus
School of Computing and Design, Sarawak, Malaysia
blau@swinburne.edu.my

Abstract. This research focuses on utilizing the biometrics recognition to trigger the speech expresser. Our selected biometric is facial expression. Though CPC have no verbal language ability, they have facial expression ability that can be interpreted to relate to their voice speech needs. However facial expression of a CPC may not be exactly identical at all times. Furthermore CPC are unique and require special speech profiles. After a thorough research in face recognition and artificial intelligence domain, neural network coupled with Gabor feature extraction is found to outperform others. A Neural Network with Gabor filters is built to train the facial expression classifiers. This research has proven successful to help CPC to express their voice speech through software with 98% successful facial recognition rate.

1 Introduction

Biometrics is concerned with the automatic recognition of individuals based on their physiological or behavioral characteristics. Among the many body characteristics that have been used, facial expression is one of the most commonly used characteristics and has been studied across a number of research fields, e.g. computer vision, pattern recognition [5, 11, 12]. Facial expression recognition is a non-intrusive method that captures human subjects' facial expressions through still images and/or video sequences in both controlled static environment and uncontrolled cluttered environment. There are two interdependent subtasks in any automated facial expression recognition that include Detection (with rough normalization) and Extraction. First, a facial expression recognition system detects a face within the captured image or video sequence; then it extracts features from potential face for recognition. Many face detection methods have been published in recent years, such as the Adaboost approach, neural network, support vector machine (SVM) and distribution-based approach [12]. Most of the methods do not provide good performance in real time applications. This is due to the scale of the faces cannot be predefined gives rise to computation complexity and time consumption. However, for our cerebral palsies, it is possible for us to capture at a fixed scale as they do not have much movement. Furthermore, we adapt the facial expression system to their wheelchairs with a fixed image capturing distance. In this research, we focus on developing a speech expression system for the severe palsies who could not form understandable speeches. Though CPC have little verbal language ability, they have the abilities to show various facial expressions. The

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 591–598, 2009.

expressions can be analyzed and interpreted to relate to their voice speech needs. However facial expression of a CPC may not be exactly identical at all times. Furthermore each CPC is unique and requires special speech profiles. We aim to understand then assist the CPC to express their desires by incorporating present facial expression recognition technology into our system. Why do we choose facial expressions? According to psychologist, facial expression provides information about emotional states as well as cognitive activities [4]. Emotions are revealed earlier through facial expression than people verbalize or even realize their emotional state [12]. Emotions can be classified under six basic categories by the renowned psychologist, Paul Ekman [4] that is fear, anger, sad, surprise, disgust, happy. Some psychologists also concluded that cognitive interpretations of emotions from facial expressions are innate and universal to all humans regardless of their culture [11, 12].

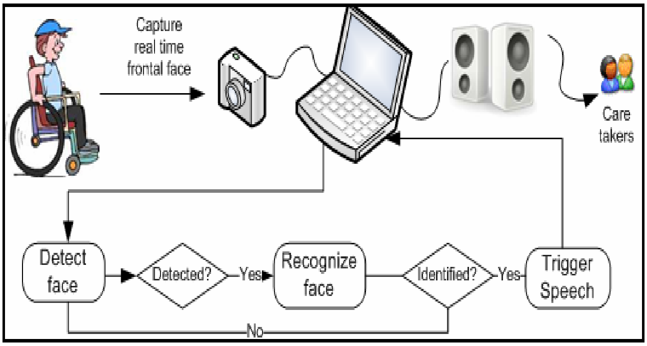


Fig. 1. Proposed system flow

2 Facial Expression Recognition

At this initial stage of development, our proposed CPC speech expression system performs under controlled indoor and outdoor scenes. We have selected a few cooperative CPC (with permissions from their guardians) to capture their proper 2D frontal images. As you can see in Fig. 1, a CPC on the wheelchair has his 2D frontal image taken from a digital camera attached to a fixed location. Ideally, the whole system should be mounted on the wheelchair to go along with the CPC. Our system does the face detection to determine whether there is a face in a given image and extract the location and extent of the face. The extracted face image is processed in real time basis with our proposed hierarchical detection scheme performing a guided coarse-to-fine search so that the system efficiency could be improved. There are two main aspects our facial expression recognition. First of all, Gabor features extraction is performed on the training images in the database, for each image of each human subject, a set of unique features are extracted. Each set of unique features is globalized into one feature vector to represent a human subject. Consequently, all the global feature vectors obtained from our human subjects are supplied as input to train a back propagation neural network. The rationale of utilizing neural network in our facial expression recognition model is due to our human subjects cannot control their facial expression to be exactly the same at all time, some variances can occur. However, our study found out that the variations are within a range that can be trained and modeled through neural network. Hence a back propagation neural network is fed with those variations for each human subject. The output of the neural network is a set of trained facial expressions and non facial expressions for each human subject. Our system starts matching a detected face from a captured 2D frontal image to the stored facial expression collections. Once a detected face is matched with a stored facial expression in our collection, a speech may be expressed through the speakers attached to our system accordingly. For instance, if a facial expression is matched to Child A's tagged expression for hungriness in the our database, then a voice representing Child A will be played through the loud speaker, "I am hungry, I want to eat now!", hence the caretakers, parents or teachers can obtain the message clearly from a far, and attend to the CPC with his food. Presently, our targeted CPC is the most severe one with very weak hand coordination skills to touch any device; however he has the ability to swing his neck and make some significant facial expression and sounds when he has a need. As for the context, the CPC are most probably being seated in a wheel chair or laid on the bed at all times as they have little physical mobility.

2.1 Face Image Preprocessing

The captured images from our human subjects are transformed into gray scale images. The centers of two eyes on each gray scale image are used as the centers for rotation, translation, scaling and cropping. Each processed image has a size of 256×256 pixels. The preprocessed images are then subject to contrast/illumination and histogram equalization. Contrast is a measure of the human visual system sensitivity. The face recognition process in different lighting conditions with different illumination and contrast has different level of efficiency and psychologically meaningfulness. Hence, for our CPC application, all images are processed with same illumination and root mean square (RMS) contrast. The RMS contrast metric is equivalent to the standard deviation of luminance [9]. x_i is a normalized gray-level value such that $0 < x_i < 1$ and x is the mean normalized gray level. With this normalization, images of different CPC faces have the same contrast as long as their RMS contrast is equal. RMS contrast does not depend on spatial frequency contrast of the image or the spatial distribution of contrast in the image. All the faces are maintained with the same illumination and same RMS contrast where α is the contrast and β is the brightness to be increased or decreased from the original image f to the new image g as in Equation 2. On the other hand, Histogram equalization is used to compensate the lighting conditions and enhance the contrast of the image. This is due to the CPC face images may encounter poor contrast because of the limitations of the lighting conditions especially indoor.

$$RMS = \left[\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right]^{\frac{1}{2}} \tag{1}$$

$$g = \alpha f + \beta \tag{2}$$

2.2 Gabor Feature Extraction from Face

Basically, we proposed to divide a face image into two portions only that is left and right face as shown in Fig. 2. Right face consists of right mouth, right cheek and

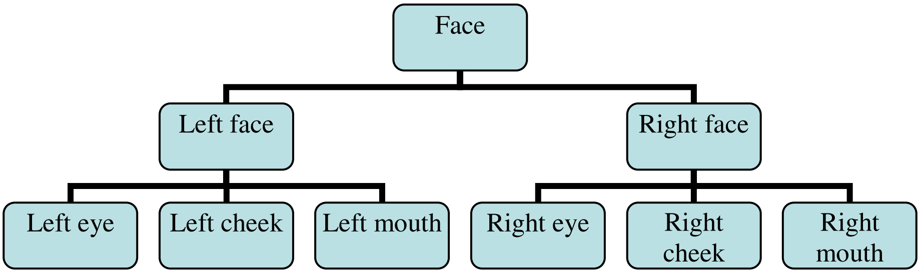


Fig. 2. Hierarchical feature extractions from a face

right eye whereby left face consists of left mouth, left cheek, and left eye. In our facial expression extraction, Principal Component Analysis (PCA) method is applied. Multiple Essential Feature Points (ESP) are located by a hierarchical component-based feature recognizer which will provide the coordinate location.

$$
\begin{aligned}
&g(x,y)=\left(\frac{1}{2\pi\sigma_x\sigma_y}\right)\exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}\right)+\left(\frac{y^2}{\sigma_y^2}\right)+2\pi jWx\right]\\
&W=U_h,\sigma_x=2\sigma_u/\pi,\sigma_y=2\sigma_v/\pi\\
&\sigma_u=\frac{(a-1)U_h}{(a+1)\sqrt{2\ln 2}}\\
&\sigma_v=\tan(\left(\frac{\pi}{2K}\right)\left[U_h-2\ln\left(\frac{\sigma_u^2}{U_h}\right)\right]\left[2\ln 2-\frac{(2\ln 2)^2\sigma_u^2}{U_h^2}\right]^{-1/2}\\
&a=(U_h/U_l)-\frac{1}{s-1}
\end{aligned}
\tag{3}
$$

$$
\begin{aligned}
\mu_{mn}&=\iint |W_{mn}(x,y)|\,dxdy\\
\sigma_{mn}&=\sqrt{\iint (|W_{mn}(x,y)|-\mu_{mn})^2\,dxdy}
\end{aligned}
\tag{4}
$$

$$W_{mn}(x,y)=\int I(x_1,y_1)g_{mn}{}^*(x-x_1,y-y_1)dx_1dy_1 \tag{5}$$

$$F_{mn}(x_F,y_F)=W_{mn}[x_F,\ldots,x_{F+s}\mid y_F,\ldots,y_{F+s}] \tag{6}$$

$$\bar{X}=[F_0,F_1,F_2,\ldots,F_{60}],\quad F=[\mu_{00},\sigma_{00},\mu_{01},\sigma_{01},\ldots,\mu_{mn},\sigma_{mn}] \tag{7}$$

From the analysis we have conducted, we found out that CPC produces significant Gabor features on the parts indicated in Fig. 2 due to their physical inability in controlling the face muscle. The location points of the eye, cheek and mouth are being derived from the location of the center, top, bottom, left and right corners respectively. A pre-defined global filter based on the two-dimensional Gabor wavelets $g(x, y)$ are defined as in Equation 3 and 4 [7]. K is the total number of orientations, and s is the number of scales in the multi-resolution decomposition. U_h and U_l is the lower and upper center frequencies respectively. The mean and standard deviation of the convolution output is used as the representation for classification. Given an image $I(x, y)$, the Gabor wavelet transformed is defined as W_{mn} in Equation 5. The localized Gabor

Feature $F(x_F\ ,\ y_F\ )$ can be expressed as a sub-matrix of the holistic Gabor wavelet output from Equation 3 where *s* defines the size of the feature area. The x_F and y_F can be defined respectively as $x_F = x_{RF} + c$ and $y_F = y_{RF} + c$ with $-20 \le c \prec 10$ where the subscript "*RF*" refers to the relative center location coordinates for the eyes, cheek and mouth as in Equation 6. The Localized Gabor Feature (LGF) vector of each of the image can be formed as in Equation 7. Each of feature point *F* is the sub-matrix of the convolution output for the image with the Gabor features bank.

3 Back Propagated Neural Network (NN)

Our facial expression recognition is achieved by employing a multilayer perceptron with back propagation algorithm as shown in Fig. 3. The architecture of the neural network is illustrated in Fig. 4. The input layer receives Gabor features detected as its input. The number of nodes in Gabor layer equals to the dimension of the feature vector incorporating the Gabor features. The number of nodes in the output layer equals to the number of individual faces the network is required to recognize. The number of epochs for this experiment was 10,000 and the goal was 0.01. The back propagated neural networking training algorithm is shown in Fig. 5. In the

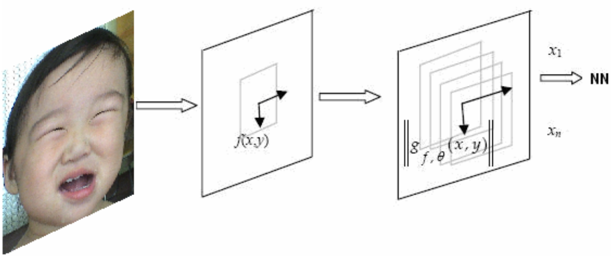


Fig. 3. Gabor features extraction Layer

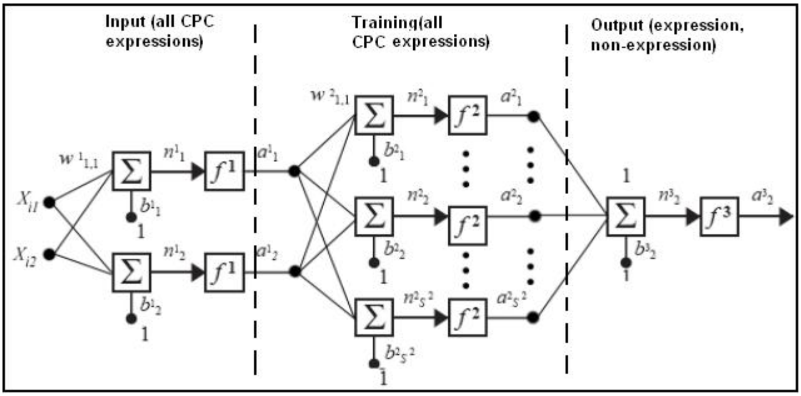


Fig. 4. Gabor features based Back Propagated Neural Network

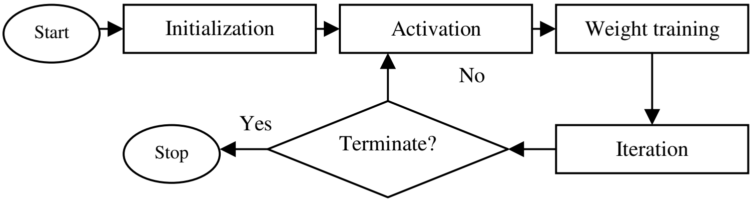


Fig. 5. Back propagation algorithm

initialization stage, all the weights and threshold values of the network are set to random numbers within (–F_i, +F_i) where F_i represents the sum of neurons, *i* in the network. In the activation stage, the network is activated by applying the inputs $x_1(t)$, $x_2(t)$,..., $x_n(t)$ and the desired outputs $y_1(t)$, $y_2(t)$,..., $y_n(t)$. The actual outputs in the training and output layers are calculated. In the weight training stage, all weights are updated, and the errors associated with the output neurons are propagated backward. Iteration, *t* is increased by 1. If termination does not occur, then the back propagation iterates again.

4 Testing and Evaluation

In order to evaluate the effectiveness of the proposed method, experiments were carried out for real images. The CPC face database is tested and discussed here in comparisons to some commercially available face database. The testing and evaluation has been performed on two face databases that are AT&T and JFFE (Japanese Female Facial Expression) before implemented on the CPC as a pilot study. AT&T has a total of 400 2D frontal face images from 40 human subjects. On the other hand JFFE has a collection of 213 2D frontal face images posed by 10 female human subjects. The AT&T database is tested first with our facial expression recognition using Gabor features based back propagated Neural Network. AT&T database contains 10 different images of 40 distinct human subjects in 5 different illumination conditions. Originally, each image is 92x112 pixels with 256 grey levels per pixels. The image is resized to 256 x 256 pixels to maintain consistency to our second face database test. For some human subjects, the images were taken at different times, varying lighting, facial expressions and facial details (glasses/no-glasses). All the images are taken against a dark homogeneous background and the subjects are in up-right, 2D frontal position with a tolerance for some side movement. This mimics the context of our CPC image setting where most of illuminations are due to sunlight and indoor lighting. The CPC are either in the care centre, home or school due to mobility restrictions. A 10% of the 400 images in the database were used as a training dataset and the remaining images were used as probe images in the facial expression recognition test. All images were subjected to Gabor filters discussed above and were convolved with Gabor filters. We used 10% that is 40 basis vectors of the 400 basis vectors representing 400 images. To each face image, the output equals to the number of individual faces the network is required to recognize which records the magnitudes of the Gabor filter response. The AT&T images are fully tested for face detection and facial expression recognition from database. The testing was performed with Pentium 4

3.00GHz CPU, 1 GB RAM, at the average running time for a face on an image to be detected and recognized (matched) is averagely 10 seconds for all the 40 human subjects tested. Table 1 shows accuracy of face recognition rate from each human subject. Averagely, AT&T achieves an average of 95.5% recognition rate. From our analysis on CPC face images, we found that CPC have more expressions through their mouth and cheek movements. The AT&T faces do not provide us with plenty of facial expressions that emphasized on these two locations on the face. Hence, the JFFE database [6] is used to validate the model is useful for more significant or stronger facial expressions. JFFE contains 213 images of 7 facial expressions posed by 10 Japanese female models. Each image has been rated on 6 emotion adjectives by 60 Japanese subjects. However we only selected 200 images by 10 models with 6 significant expressions each. We create a speech expression to relate to each model's facial expression. Hence the right match of facial expression will trigger speeches from the sound database to signal caretakers of a CPC's feelings or demands. The average recognition rates from the JFFE, 96.5% is better than what we have obtained in the testing of AT&T database. The main reason behind this is the facial expressions for each human subject are more distinctive causing less mismatched. However we do take into account the lack of gender difference in JFFE as it consists of all females. Furthermore, the database contains fewer images, 200 as compared to 400 2D frontal images. Table 1 also summarizes the 40 CPC face images we have. Obviously, our proposed Gabor features that focused on eye, mouth and cheek have achieved higher success rate on CPC face images as compared to ordinary human subjects in AT&T and JFEE. This shows that by utilizing the main features from eye, cheek and mouth is sufficient to detect the facial expressions on CPC.

Table 1. Facial expression recognition results for 200 images from JFEE

	AT&T		**JFEE**		**CPC**	
Facial expressions class	Average recognition rate (%)	Standard Deviation	Average recognition rate (%)	Standard Deviation	Average recognition rate (%)	Standard Deviation
Hungry	96.5	0.48	97.1	0.50	98.5	0.51
Like	96.1	0.54	96.5	0.52	99.0	0.53
Dislike	95.8	0.52	96.1	0.49	97.8	0.49
Fear	96.5	0.46	97.0	0.50	98.5	0.48
Tired	97.1	0.47	96.2	0.51	98.8	0.49
Bored	95.6	0.51	96.1	0.51	98.1	0.53
Mean	**96.3**	**0.5**	**96.5**	**0.51**	**98.5**	**0.51**

5 Conclusion

This research has developed a facial expression recognition system for severe CPC's speech expression and communication purpose. Two commonly used face databases i.e. AT&T and JFFE are used in our research due to difficulty in getting sufficient cooperative CPC. Through our testing, we found out that the proposed system that uses facial expression recognition to trigger speech for communication is feasible. Our proposed model has achieved an average of 96.4% recognition rate for the 600 images we tested. When applied to the four CPC, the facial expression recognition has

98.5% success rates. Our CPC have difficulty in the muscle coordination which mainly on either left or right of the face. Our proposed feature points to be used in the CPC are lesser than for the ordinary human subjects.

References

[1] AT&T Face Database (The ORL Face Database), http://www.cl.cam.ac.uk/Research/DTG/attarchive/pub/data/att_faces.zip

[2] Boumbarov, O., Sokolov, S., Gluhchev, G.: Combined face recognition using wavelet packets and radial basis function neural network. In: CompSysTech 2007: Proc. of International conference on Computer systems and technologies, ACM Press, New York (2007)

[3] Ekman, P.: Facial Expressions. In: Dalgleish, T., Powers, M. (eds.) Handbook of Cognition and Emotion. John Wiley & Sons Ltd., Chichester (1999)

[4] Ekman, P.: Emotions Revealed. First Owl Books. Henry Holt and Company LLC, New York (2004)

[5] Heisele, B., Ho, P., Wu, J., Poggio, T.: Face recognition: component based versus global approaches. Computer Vision and Image Understanding 91(1/2), 6–21 (2003)

[6] Japanese Female Facial Expression Database, http://www.kasrl.org/jaffe_download.html.

[7] Manjunath, B.S., Ma, W.Y.: Texture Features for Browsing and Retrieval of Image Data. IEEE Transactions on Pattern Analysis and Machine Intelligence 18(8), 837–842 (1996)

[8] Mohabbati, B., Kasaei, S.: An efficient wavelet network based face detection algorithm. In: The First IEEE and IFIP International Conference in Central Asia. IEEE, Los Alamitos (2005)

[9] Weyrauch, B., Huang, J.: Component-based Face Recognition with 3D Morphable Models. In: proceedings of 4th Conference on Audio- and Video-Based Biometric Person Authentication, pp. 27–34 (2003)

[10] Wong, J.J., Cho, S.Y.: Facial emotion recognition by adaptive processing of tree structures. In: Proc of ACM symposium on Applied computing. ACM Press, New York (2006)

[11] Zana, Y., Roberto Jr., M.C.: Face recognition based on polar frequency features. ACM Transactions on Applied Perception (TAP) 3(1) (2007)

[12] Zhao, W., Chellappa, R., Phillips, P.J., Rosenfeld, A.: Face recognition: A literature survey. ACM Computing Surveys (CSUR) 35(4) (2003)

Investigations into Particle Swarm Optimization for Multi-class Shape Recognition

Ee Lee Ng, Mei Kuan Lim, Tomás Maul, and Weng Kin Lai

Centre for Multimodal Signal Processing,
MIMOS Bhd., Malaysia
{lee.ng,lim.mkuan,thomas.henrique,lai}@mimos.my

Abstract. There has been a significant drop in the cost as well as an increase in the quality of imaging sensors due to stiff competition as well as production improvements. Consequently, real-time surveillance of private or public spaces which relies on such equipment is gaining wider acceptance. While the human brain is very good at image analysis, fatigue and boredom may contribute to a less-than-optimum level of monitoring performance. Clearly, it would be good if highly accurate vision systems could complement the role of humans in round-the-clock video surveillance. This paper addresses an image analysis problem for video surveillance based on the particle swarm computing paradigm. In this study three separate datasets were used. The overall finding of the paper suggests that clustering using Particle Swarm Optimization leads to better and more consistent results, in terms of both cluster characteristics and subsequent recognition, as compared to traditional techniques such as K-Means.

1 Introduction

Security surveillance systems are becoming very common nowadays especially in public spaces where personal safety is of utmost importance [1]. Conventional security surveillance systems require the constant attention of security personnel, to monitor several locations concurrently [2][3]. This is a difficult and error-prone task in dire need of automation. Hence, the advancement of image processing techniques has become an important tool in the video surveillance system's arsenal where they can help improve the operational aspects of monitoring through closed circuit cameras.

One of the basic building blocks of such systems consists of the ability to identify and classify patterns of interest (e.g. humans, weapons and suspicious behaviours). In multi-class shape recognition, the task is to classify the feature space into more than two regions using a set of discriminate functions, in which each region corresponds to a pattern class. More specifically, given n vectors of instances $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ drawn from feature space Ω, a multi-class classifier has to classify the inputs into k pre-defined classes $\boldsymbol{C} = (C_1, C_2, \ldots, C_k)$, where $C_a \neq C_b$ for $a \neq b$ and $k > 2$. A number of studies have been carried out in the area of multi-class classification [4][5][6].

There are two main approaches to image classification: supervised learning (classification) and unsupervised learning (clustering). Clustering divides objects into clusters based on the mutual similarity of objects [7]. Each cluster then contains patterns representing objects that are similar according to the selected object description and similarity criteria (features).

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 599–606, 2009.

Our approach to do this is based on a population based stochastic optimization approach, namely particle swarm optimization [8] (*PSO*). In this paper, we present a standard *PSO* implementation to cluster objects into their appropriate classes. The experimental results indicate that *PSO* is capable of delivering better clusters as compared to a conventional approach known as *K-Means*.

The rest of the paper is organized as follows. Section 2 introduces the experimental setup. Section 3 describes and briefly discusses the results and finally section 4 gives the conclusion of the entire study.

2 Particle Swarm Optimization (PSO)

Particle Swarm Optimization or *PSO* was developed by *Eberhart* and *Kennedy* in 1995 after they were inspired by the social behaviour of a flock of birds [8]. In the *PSO* algorithm, the flock of birds is symbolically represented as a set of particles and these particles represent agents flying and searching through a problem space for the best solution. A particle's location in the multi-dimensional problem space represents one solution to the problem. When a particle moves to a new location, a different solution is generated and will be compared with the other particles. The interim solution achieved is then evaluated by using a fitness function where it provides a measurable quantitative value which will range from 0 to 1. Theoretically, after all of the particles have been compared with one another by using this fitness function, the particle with the best solution will be the reference where all the other particles refer to and "*move*" towards, while searching for other better solutions at the same time.

Technically, the *Particle Swarm Optimization* algorithm may be represented by equations (1) and (2):

$$V_{id} = K*[V_{id} + (C_1*rand_1*(P_{id} - X_{id})) + (C_2*rand_2*(P_{gd} - X_{id})) \tag{1}$$

where

$$X_{id}^{new} = X_{id}^{old} + V_{id} \tag{2}$$

and

$$K = \frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|}, \text{ where } \varphi = c_1 + c_2,\ \varphi > 4 \tag{3}$$

V_{id} determines the velocity of a swarm and therefore controls the movement and the updating magnitude for a particle. X_{id}, in equation (2) is used to update the current position of a particle to a new position after it "moves". When the best location of the particle is found, this location will be stored away as $P_{best..}$ On the other hand, P_{gd} indicates the best location from all the best locations found by each individual particle and is normally denoted as G_{best} or the *global* best. C_1 and C_2 are acceleration coefficients whereas $rand_1$ and $rand_2$ are random values that vary from 0 to 1 [8].

3 Experimental Setup

Three different data-sets were used to test the *PSO* clustering technique. The first one came from the popular Iris flower dataset [9]. This basically contains 150 samples of measurements from the Iris flower consisting of a total of 4 features each with two measurements taken from the petals and sepals.

The second data-set used is somewhat larger and is based on the features extracted from four separate classes of objects (human, animal, luggage, and vehicle) which make up the key objects in the video surveillance system that we are developing.

The third data-set, consists of 193 shapes in 8 categories. The dataset was selected from [10] and the MPEG-7 test database. Unlike the previous data-sets, this data-set was characterized as *imbalanced* because some of the classes are represented by a significantly larger number of instances as compared to the other classes.

Our implementation of *PSO* was typically run with 10 randomly initialized particles and was defined to stop after 300 iterations. It is common in the literature for 10 to 60 particles to be used. The *PSO* algorithm was run 30 times in order to characterize its performance. At the end of each run, the best fitness value was chosen. The main fitness function being optimized by our *PSO* implementation was:

$$f(x_i, Z_i) = \frac{\bar{d}_{\max}(Z_i, x_i) + J_{e,i}}{d_{\min}(Z_i, x_i)} \tag{4}$$

where,

$$J_e = \frac{\sum_{k=1}^{K} \left[\sum_{\forall z_p \in C_k} d(z_p, m_k) \right] / n_k}{K} \tag{5}$$

where x_i represents a particular *PSO* particle (i.e. a set of centroids), Z_i consists of a matrix representing the pattern-to-cluster assignments for particle *i*, $\bar{d}_{\max}$ represents the maximum average Euclidean distance of patterns to their associated cluster centroids, $d_{\min}$ refers to the minimum total Euclidean distance of the data to its respective cluster centroid, C_k represents the kth cluster, n_k refers to the number of objects that belong to a particular cluster *k,* and the function *d* denotes the distance between two arguments (e.g. the distance between a cluster pattern and the cluster's centroid). This fitness function is a useful measurement of the quality of the clustering algorithm in terms of compactness.

As already mentioned, the performance of our *PSO* implementation was compared with *K-Means* [11]. This is a well-known clustering technique which will cluster *n* objects based on attributes into *k* partitions, where *k* < *n*. It is similar to the expectation-maximization algorithm for mixtures of Gaussians in that they both attempt to find the centers of natural clusters in the data. It assumes that the object attributes form a vector space. The objective it tries to achieve is to minimize total intra-cluster variance, or, the squared error function, viz.

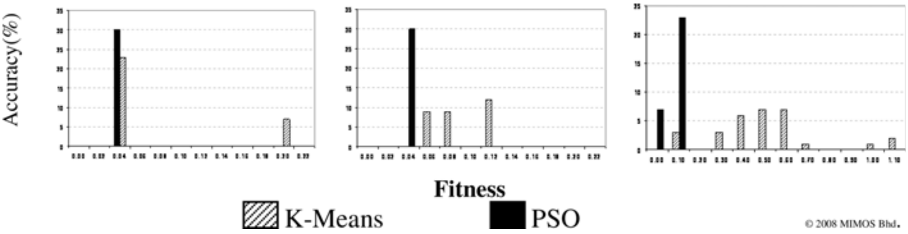
$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} (x_j - \mu_i)^2 \tag{6}$$

where there are *k* clusters S_i, i = *1, 2, ..., k,* and μ_i is the centroid or mean point of all the points $x_j \in S_i$.

4 Results and Discussion

The quality of the clusters generated by the *PSO* algorithm is based on the overall fitness value as determined by Equations 4 and 5. The fitness value can also give a quantitative measure of each cluster's compactness. The smaller the fitness value, the more compact the clusters and vice versa. Over time, *PSO* is capable of generating more compact clusters as the fitness value can be seen to decrease.

The charts in Fig. 1 show the results for both *PSO* and *K-Means* for the 3-class, 4-class and 8-class data-sets. The values reported here are taken over a total of 30 simulations for each clustering technique. As a partitional clustering technique, *K-Means* tends to converge faster but usually produces a range of clusters since it tends to locate centroids based on the initial random cluster configuration. In contrast, the clusters obtained by *PSO* were of consistently lower fitness values.



© 2008 MIMOS Bhd.

Fig. 1. From left to right the charts depict performance measures for the 3-class, 4-class and 8-class data-sets respectively. The x-axes consist of fitness values, and the y-axes consist of the numbers of solutions found for particular fitness values.

Table 1 summarizes the results obtained by each of these clustering techniques. For the two balanced data sets (i.e. 3-class and 4-class), *PSO* was able to search through the solution space to come up with clusters which are far better than *K-Means*. Here, even the worst clusters are better than those found by *K-Means*. Now even though *PSO* was able to generate better clusters than *K-Means*, on the average, the best solutions found by the latter are comparable to those from *PSO*.

Another measure of the quality of the clusters is the inter- and intra-cluster distances. The intra-cluster distance ensures compact clusters with little deviation from the cluster centroids, while the inter-cluster distance ensures larger separations between the different clusters. The inter- and intra-cluster distances are summarized in Table 2.

Table 1. Total Quantization Error for each Clustering Technique

Data	***K-Means***			***PSO***		
	Min	**Avg**	**Max**	**Min**	**Avg**	**Max**
3-class	0.054209	0.091321	0.213260	0.040000	0.040301	0.042231
4-class	0.065390	0.094999	0.125980	0.040002	0.041477	0.046155
8-class	0.134830	0.554610	1.172200	0.091212	0.107645	0.199400

(© 2008 MIMOS Bhd.).

Table 2. Cluster quality comparisons for K-Means and PSO.

Data	***K-Means***		***PSO***	
	Intra-cluster	*Inter-cluster*	*Intra-cluster*	*Inter-cluster*
3-class	0.0597± 0.0130	0.6730± 0.0687	0.4677±0.3958	2.0537±1.6064
4-class	0.0641±0.0016	0.5478±0.0545	0.9026±0.7312	3.6482±2.9789
8-class	0.0256±0.0130	0.7980±0.2423	0.1049±0.0623	1.3792±0.6134

With reference to this particular criterion, *PSO* was capable of clustering patterns such that there are significantly larger separations between them. Nevertheless, it seems that the clusters occupy a larger space as indicated by the intra-cluster distances obtained.

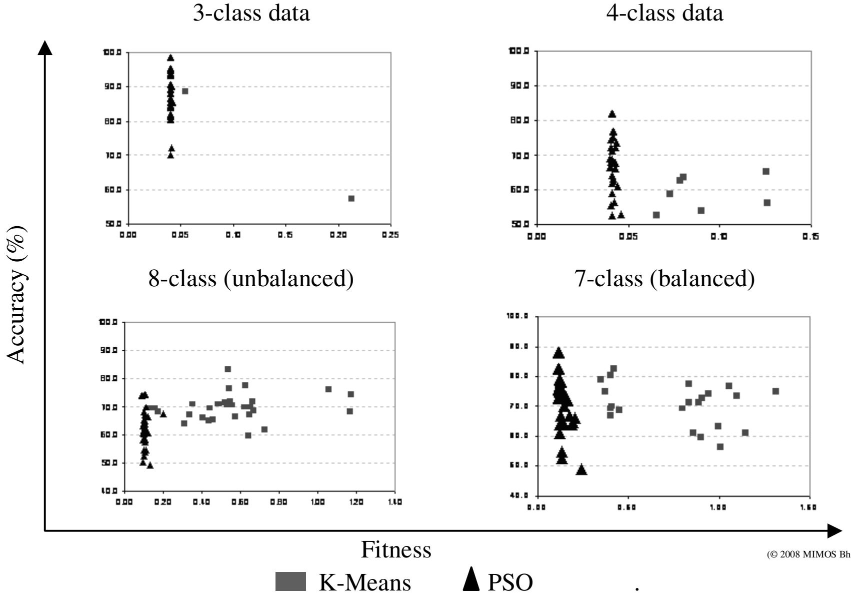
Another quality metric is to adopt a "visual inspection" for each and every object (or image) that has been grouped together within the cluster. This would give us another indication of how well the data has been clustered. This is possible because we do in fact know the class labels beforehand. Fig. 2 summarizes how well *PSO* and *K-Means* have grouped each of the objects from the perspective of accuracy.

The results indicate that *PSO* is able to generate clusters which are significantly better compared to *K-Means*, especially when the clusters are balanced (Fig. 2, top row). We know that there is significant class imbalance in the third data set consisting of 8-classes[1]. Here the performance of *PSO* is significantly poorer when compared with the clusters identified by *K-Means* (bottom-left chart in Fig. 2).

As a subsequent experiment, we wanted to study the performance of *PSO* on the third data-set when the class data is better balanced. However, if we were to have a balanced data set here, it would mean that each class would only contain 12 objects, meaning that from an initial total of 193, the data set would now contain 96 instead. Alternatively, we removed class #5 thereby allowing each class to contain only 19 images, to generate a 7-class data set. The results are shown in the bottom-right chart of Fig. 2. The reason why we chose to remove class #5 as opposed to any other class, was that this led to the overall maximum number of samples in the data-set.

We also experimented with re-balancing the third data-set by adding several new samples and removing others such that each class was represented by 19 samples (total of 152 samples). The resulting average *PSO* fitness value was 0.151942, which represents a significant improvement relative to the average *K-Means* fitness value of 0.752349. On the downside, the average *PSO* accuracy, compared to *K-Means*, was slightly lower, i.e.: 72.6% vs. 74.1%. The maximum accuracy was also slightly lower for *PSO*, i.e.: 88.8% vs. 90.1%. The average inter and intra-cluster distances were larger for *PSO* (1.46 and 0.14) compared to *K-Means* (0.81 and 0.019). Table 3 shows the minimum, maximum and average fitness values, for the two approaches.

[1] Class #1 to class #8 are known to contain 50, 19, 33, 20, 12, 20, 19 and 20 samples respectively.



(© 2008 MIMOS Bhd.)

Fig. 2. Fitness versus accuracy for all data-sets. The x-axes for all charts consist of fitness values and the y-axes consist of accuracy values.

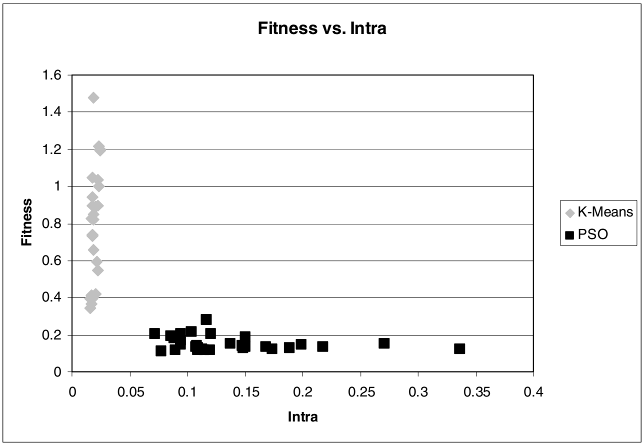
Table 3. K-Means and PSO fitness values for the 19 samples per class 8-class data-set

8 Classes Balanced (19 samples per class)					
K Means			***PSO***		
Min	**Avg**	**Max**	**Min**	**Avg**	**Max**
0.345950	0.752349	1.477800	0.108960	0.151942	0.279690

(© 2008 MIMOS Bhd.).

Fig. 3 illustrates an interesting contrast between the best solutions generated by both approaches. On one side, *K-Means* produces solutions which tend to exhibit low average intra-cluster distances, but which exhibit a large variability of fitness values, whereas on the other side, *PSO* produces the converse pattern, i.e.: solutions with a large variability of average intra-cluster distances which nevertheless tend to have low fitness values. This contrast is most likely due to the fact that both approaches are optimizing different cost functions.

As a final experiment, we allowed *PSO* to optimize a fitness function more closely related to the one implicitly being optimized by *K-Means* (i.e. intra-cluster variance). To be more precise the fitness function we selected essentially consisted of the mean of the sum of intra-class distances for each cluster. When optimizing this function, even when applied to the original unbalanced 8-class data-set, *PSO* was found to outperform *K-Means*. Table 4 provides us with a concrete idea of the average, best and worst performances for each approach, using this fitness function, on all of the three original data-sets. It is clear here, that optimizing intra-cluster variance using *PSO* is superior to implicitly doing this via a traditional *K-Means* implementation. It



(© 2008 MIMOS Bhd.).

Fig. 3. The relationship between average intra-cluster distances and fitness values for both K-Means and PSO

Table 4. Performance (accuracy) of the intra-cluster variance function on the three data-sets

Data Set	***K-Means***			***PSO***		
	Min	**Avg**	**Max**	**Min**	**Avg**	**Max**
3-class	57%	80%	89%	57%	86%	90%
4-class	52%	60%	65%	50%	60%	75%
8-class	49%	65%	77%	62%	74%	87%

(© 2008 MIMOS Bhd.).

is interesting to note that, for the 8-class data-set, the correlation between fitness and accuracy values (data not shown) for the *K-Means* case was -0.9044 while that for *PSO* was -0.25813. It should be interesting to investigate the reason behind *PSO*'s relatively weak correlation, in this case.

5 Conclusions and Further Work

In this paper, we have applied a population based stochastic optimization technique to perform unsupervised clustering of multi-class images. The results were compared with a well known clustering technique known as *K-Means*. Unlike *K-Means*, *PSO* is an optimization technique inspired by swarming phenomena found in nature. Based on the cost function defined in equations (4) and (5), our standard *PSO* implementation was found to perform significantly better than *K-Means*. However, we also compared the resulting fitness values to their corresponding accuracy values and noted that clearly, there were further differences in terms of cluster memberships and correlation patterns between fitness and accuracy that warranted further investigation.

The paper has revealed the potential of *PSO* for its ability to recognize multi-class images. Nevertheless, it does not seem to perform quite as well for imbalanced class data-sets. This problem was shown to be rectified by using a fitness function which is closer to what *K-Means* is implicitly optimizing, as Table 4 demonstrates. In the future, we would like to conduct a thorough investigation into how *PSO* performs with different fitness functions. One interesting question pertains to which functions lead to larger accuracy variances for individual fitness values.

Acknowledgements. The authors wish to acknowledge *Li Zhi Yuan, Wong Chin Pin* and *Cheong Kee Loan* for their contributions at the early stages of this piece of work.

References

1. Ali, A.T., Dagless, E.L.: Computer vision for security surveillance and movement control. IEE Colloquium on Electronic Images and Image Processing in Security and Forensic Science, 1–7 (1990)
2. Fong, A.C.M.: Web-based intelligent surveillance systems for detection of criminal activities. Journal of Computing and Control Engineering 12(6), 263–270 (2001)
3. Chien, Y.T., Huang, Y.S., Jeng, S.W., Tasi, Y.H., Zhao, H.X.: A real-time security surveillance system for personal authentication. In: IEEE 37th Annual 2003 International Carnahan Conference on Security Technology, pp. 190–195 (2003)
4. Ou, G., Murphey, Y.L.: Multi-class pattern classification using neural networks. Pattern Recognition 40(1), 4–18 (2007)
5. Schwenker, F.: Solving multi-class pattern recognition problems with tree-structured support vector machines. In: Proceedings of the 23rd DAGM-Symposium on Pattern Recognition, pp. 283–290 (2001)
6. Weston, J., Watkins, C.: Support vector machines for multiclass pattern recognition. In: Proceeding of the 7th European Symposium On Artificial Neural Networks (1999)
7. Sonka, I., Hlavac, V., Boyle, R.: Image Processing, Analysis and Machine Vision (Thomson Learning 2008)
8. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of the IEEE International Joint Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
9. Fisher, R.A.: The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics 7, Part II. 179-188 (1936)
10. Sebastian, T.B., Klein, P.N., Kimia, B.B.: Shock-based indexing into large shape databases. In: Proceedings of 7th European Conference on Computer Vision, pp. 83–89 (2002)
11. Steinhaus: Sur la division des corp materiels en parties. Bull. Acad. Polon. Sci. C1. III 4, 801–804 (1956)

Patterns of Interactions in Complex Social Networks Based on Coloured Motifs Analysis

Katarzyna Musial[1], Krzysztof Juszczyszyn[1], Bogdan Gabrys[2],
and Przemysław Kazienko[1]

[1] Wroclaw University of Technology
katarzyna.musial@pwr.wroc.pl, krzysztof.juszczyszyn@pwr.wroc.pl
[2] Bournemouth University
bgabrys@bournemouth.ac.uk, przemyslaw.kazienko@pwr.wroc.pl

Abstract. Coloured network motifs are small subgraphs that enable to discover and interpret the patterns of interaction within the complex networks. The analysis of three-nodes motifs where the colour of the node reflects its high – white node or low – black node centrality in the social network is presented in the paper. The importance of the vertices is assessed by utilizing two measures: degree prestige and degree centrality. The distribution of motifs in these two cases is compared to mine the interconnection patterns between nodes. The analysis is performed on the social network derived from email communication.

1 Introduction

The investigation of the communication patterns in the complex social networks is a very resource-consuming task. The methods that are quite useful and effective in small and medium sized social networks fail while applying them to the complex networks. In the case of large networks the solution to the complexity problem, which occurs when the models of interaction are studied, can be the analysis of the local network structures, also known as network motifs. Motif analysis stems from bioinformatics and theoretical biology [10],[13], where it was applied to the investigation of huge network structures like transcriptional regulatory networks, gene networks or food webs [7],[8]. Although the global topological organization of metabolic networks is well understood, their local structural organization is still not clear. At the smallest scale, network motifs have been suggested to be the functional building blocks of network biology. So far several interesting properties of large biological network structures were reinterpreted or discovered with help of motif analysis [14],[16].

In this work we apply this biologically-inspired set of methods to the analysis of email-based social network of the size similar to many networks observed in nature. Motif analysis offers low computational overhead and opportunity to gain an insight into the local structure of huge networks which otherwise would require prohibitive computations to investigate. Moreover, we go one step beyond the classic motif analysis and propose distinguishing network nodes with respect to their unique properties, in this case centrality values (as defined in social sciences). The discovered motifs and their numbers enable to assess which patterns of communication appear

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 607–614, 2009.

often in the large social networks and which are rather rare. The former ones can be seen as these which come into existence in a natural way whereas the latter ones can be treated as the artificial and unnatural. Moreover, the nodes of these subgraphs can be divided into different classes based on the values of various kinds of measures used in social network analysis. These additional information can be used in order to mine new knowledge from the data about interaction between users. The outcomes of the research on two–coloured, three–nodes network motifs (triads) detection and analysis in large email–based social network of the Wroclaw University of Technology (WUT), consisting of over 5,700 nodes and 140,000 edges are presented in this paper. The colours were assigned to the vertices in such a way that the black colour of the node reflects its low centrality whereas the white colour its high centrality. The frequency of occurrence and the distribution of the individual motifs serve as the basis to define the interaction patterns between users. Moreover, depending on the values of centrality measures, we can investigate in what kind of motifs (patterns of relations) people are embedded.

2 Related Work

2.1 Network Motifs

Complex networks, both biological and engineered, have been shown to display so–called network motifs [10]. They are small (usually 3 to 7 nodes) subgraphs which occur in given network far more/less often then in corresponding random networks. In order to evaluate the distribution of motifs their concentration is measured for a set of random networks then compared with the network being investigated. For computational complexity reasons, the size of the random network set should be as small as possible. For example, our former research has revealed that 100 random networks provide sufficient accuracy of calculations in the case of the WUT social network [5]. The statistical significance of a given motif is defined by its *Z*–score Z_M:

$$Z_M = \frac{n_M - \left\langle n_M^{rand} \right\rangle}{\sigma_M^{rand}}, \tag{1}$$

where: n_M is the number of occurrences of motif M in the network, $\left\langle n_M^{rand} \right\rangle$ and σ_M^{rand} are the mean and standard deviation of its appearances in the set of random networks [4]. The calculated for each motif *Z*–score measure forms the basis to create the significance profile (*SP*) of the network. It has been recently shown that distribution of network motifs may help to distinguish and classify complex biological, technical and social networks [9]. Each class of these networks has its own specific significance profile. Two methods of motif detection can be utilized. The first one assumes exhaustive enumeration of all subgraphs with a given number of nodes in the network. Note that, their computational cost dramatically increases with the network size. The second one is to use random sampling to effectively estimate concentrations of network motifs. The algorithm presented in [6] is asymptotically independent of the network size and allows fast detection of motifs in very large networks (hundreds of thousands of nodes and larger).

Network motifs can be used to describe both topological and functional properties of various networks. For biological networks it was suggested that network motifs play key information processing roles [13]. For example, so–called Feed–Forward Loop (motif number 5 on Fig. 1) has been shown both theoretically and experimentally to perform tasks like sign–sensitive filtering, response acceleration and pulse–generation [7]. Such results show that, generally, we may reason about function and properties of very large networks from their basic building blocks [8]. In another work by Chung-Yuan at al. motif analysis was proved to have the ability of fast detection of the small–world and clustering properties of a network [3]. Within the area of computer science and social networks very little work has been done with motifs. In [9] SPs for small (below 100 nodes) social networks were demonstrated. Counting $3.5\text{x}10^5$ nodes WWW network described in [2] was used to show the effectiveness of sampling algorithm in [6]. First results for large e-mail based network were presented in [5].

2.2 Centrality Measures in Social Network

Both the nodes and the edges of the network motifs can be divided into different classes (colours) based on the freely chosen measure. In this research the nodes will be coloured with respect to their centrality. Two methods – degree centrality and degree prestige – will be utilized to colour the vertices in the e-mail social network in the process of triads detection. Degree centrality $DC(x)$ of a member x takes into account the number of outdegree of member x [11], [12]. It means that a node is more important than another one when it communicates with greater number of network members. It is usually expressed by the number of neighbours that are adjacent to the given person. The degree prestige is based on the indegree number so it takes into account the number of members that are adjacent to a particular member of the community [15]. In other words, more prominent people are those who received more nominations from members of the community [1]. The degree prestige $DP(x)$ of a member x is the number of members from the first level neighbourhood that are adjacent to x.

3 Coloured Motifs Analysis

3.1 Data Preparation and Plan of the Experiments

The experiments were carried out on the logs from the Wrocław University of Technology (WUT) mail server, which contain only the emails incoming to the staff members as well as organizational units registered at the university. Based on the information about the communication between the employees at the WUT the e-mail based social network was extracted. In such a network the nodes are unique e-mail addresses and the edges reflect the fact that there exists any communication between two distinct e-mail addresses. First, the data cleansing process was executed. Only emails from and to the WUT domain were left. After that the centrality of vertices was assessed. In order to perform that task, two methods were utilized: degree centrality and degree prestige which were calculated as the number of respectively outgoing and incoming edges from or to the given node. In the next phase, the values of each calculated centrality measure were assigned to one of two classes. These classes were created based on the mean value of the given centrality measure evaluated for WUT social network. In both cases of degree centrality and degree prestige it was 25 (see Table 1). Note that around 70% of WUT employees have the degree centrality lower than the mean value and around 62% have the degree prestige lower than the mean value (Table 1).

Table 1. The number of nodes in each of the created classes

	Degree Centrality *DC*(*x*)		Degree Prestige *DP*(*x*)	
	DC(*x*)>25	*DC*(*x*)≤ 25	*DP*(*x*)>25	*DP*(*x*)≤ 25
No. of nodes	1736	4047	2165	3618
% of all nodes	30.02%	69.98%	37.44%	62.56%

The further stages of experiments were divided into two parts. Each of the parts is concerned with one of the centrality measure. To each network node one of two colours: white or black, was assigned. The black colour is assigned to the vertices that degree centrality is lower or equal 25 whereas the white colour to vertices with degree centrality higher than 25. The analogous procedure was applied in the case of the degree prestige. After that the process of detecting triads within the WUT email social network was performed. Basically, there are 13 different motifs that consist of three nodes each (Fig. 1). Their ID=1,2,…,13 are used in the further descriptions interchangeably with the corresponding abbreviations M1, M2,…, M13. The main part of the experiments took into consideration motifs with two coloured vertices, in consequence for each of the 13 motifs four classes of motifs can be enumerated i.e. motifs with all vertices white, motifs with two vertices black and one white, with two vertices white and one black, and finally three black vertices. As an example see Fig. 2, where M5 is presented together with all possible motifs that have two-coloured vertices and have been built upon the M5. The outcomes of experiments, the goal of which was to detect the described two-colored motifs and determine the role of the nodes with high DC and DP in the social network, are presented in the Sec. 3.3.

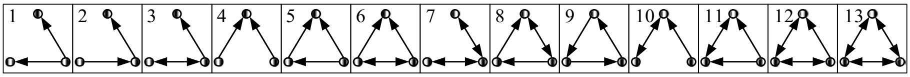


Fig. 1. Directed triads and their IDs that can exist within the social network

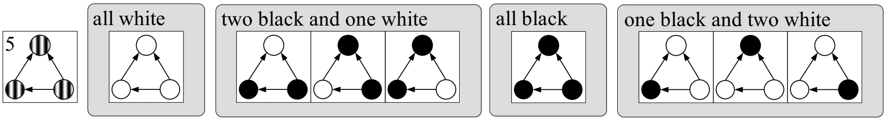


Fig. 2. Classes of the motifs

3.2 Distribution of Network Motifs in WUT Social Network

Triad Significance Profile (TSP) for WUT network presented in Fig. 3 was computed using the set of 1000 random networks. The considered network reveals the typical property of social networks – the small–world phenomenon. Loosely connected motifs with only 2 edges, like M2, M3, M4, M7 and M10 occur less frequently in comparison to the random networks. As expected, it shows a high clustering level, i.e. high probability that two neighboring nodes have connected neighbors. The only exception is 1 which occurs relatively often. This reflects specific property of large mail–based social network: there are relatively many broadcasting nodes which spread messages (news, announcements, bulletins) which are never answered.

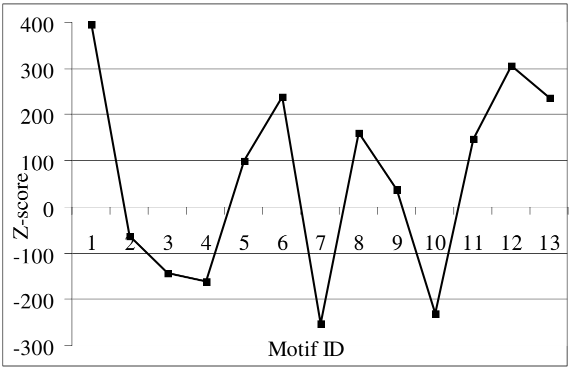


Fig. 3. Triad Significance Profile (Z-score values) of the WUT email-based social network for 1000 numbers of random networks

3.3 Analysis of the Coloured Network Motifs

The very interesting fact can be noticed while observing the motifs where all nodes are black, i.e. all users have low centrality. Their number is minimal (Fig. 4 and 5), although almost 70% of people have low degree centrality and 62% low degree prestige. Based on these results we may infer that there are no sparse structures and loosely connected cliques in the investigated network – nodes connect always with network hubs, direct communication between low-centrality nodes is present, but it does not shape the local structure of the network.

Table 2. Distribution of coloured network motifs for degree centrality

Motif ID	1	2	3	4	5	6	7	8	9	10	11	12	13
3 white	12,8	26,4	26,5	18,9	23,3	18,4	60,6	64,6	59,4	60,3	64,2	69,5	79,0
2 white 1 black	44,6	57,5	57,9	56,1	62,8	73,3	33,0	29,7	31,9	33,0	29,6	26,6	17,9
1 white 2 black	42,4	15,4	15,3	22,3	13,4	7,7	6,1	5,3	7,9	6,4	6,0	3,6	2,9
3 black	0,2	0,7	0,3	2,6	0,5	0,6	0,2	0,5	0,8	0,4	0,2	0,3	0,3

Interesting conclusions come from comparison of topological neighbourhood of the hubs (nodes with high DP and DC). When we compare the motif distribution for DC and DP (Fig. 4 and 5) it can be noticed that in case of DC the mixed motifs, i.e. these with two nodes in one colour and one in another colour, occur more often, especially in the case of motifs containing two edges (i.e. with nodes belonging to different clusters). In this way high-DC nodes ("DC hubs") tend to link low-centrality nodes with the core network, which is generally formed by hubs with high DP – 37% of network nodes with high DP constitute (alone) up to 50% of all network triads the effect of which is not seen in the case of DC (30% occurrence, forming up to 25% of all triads). This is an important conclusion suggesting that dense cliques in social network are bridged and created via nodes with high DP (more often than in the case of these with high DC), communicating with each other.

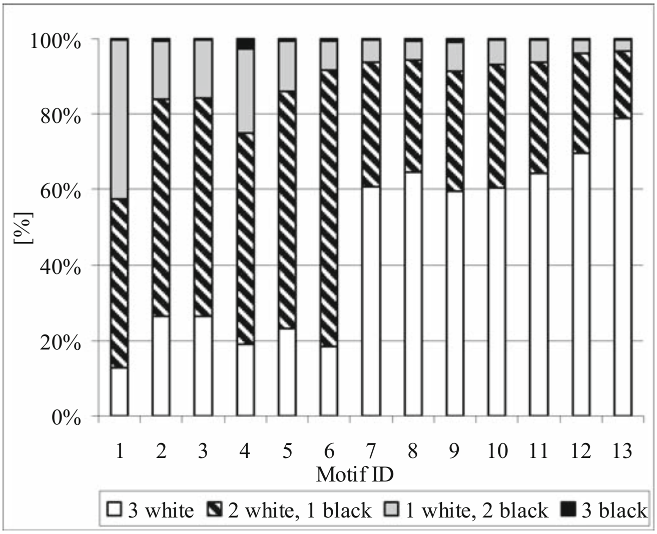


Fig. 4. Distribution of coloured network motifs for degree centrality

The bridging effect of DP hubs is indicated by high percentage of white (3 white nodes) motifs. Even taking into account that the sets of nodes with high DC and DP partially overlap (there are obviously nodes which share high DC and DP) the prevailing role of DP hubs in local topology is clearly visible.

In the case of DP, for almost all (excluding M1) motifs at least half of them consist of only white nodes, i.e. of people who have the prestige higher than the mean value for the network. Note also that, the more connected the motif is the larger number of motifs with three white nodes occur. This is true for both DP and DC (see Table 3 and 4). For example for both DP and DC three white nodes motifs constitute 80% occurrences of fully connected motif M13. It shows that the important people tend to create cliques within the social networks whereas people who are at the periphery of the network and are less important do not interact with each other.

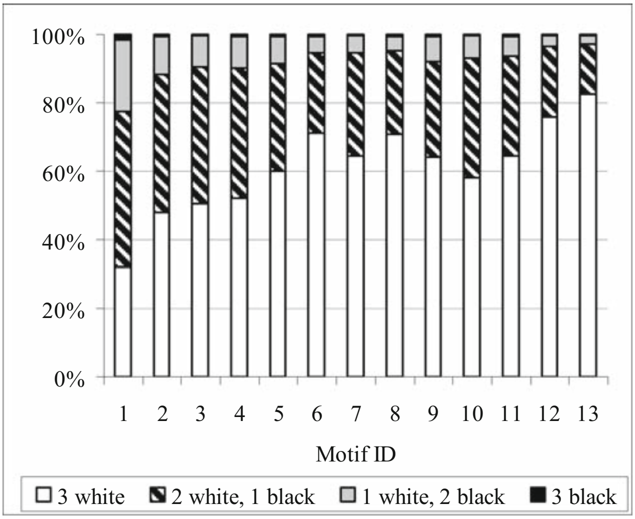


Fig. 5. Distribution of coloured network motifs for degree prestige

Table 3. Distribution of coloured network motifs for degree prestige

Motif ID	1	2	3	4	5	6	7	8	9	10	11	12	13
3 white	31,9	48,0	50,4	52,0	60,0	71,0	64,3	70,8	64,2	58,1	64,5	76,0	82,6
2 white 1 black	45,5	40,3	40,0	38,0	31,5	23,8	30,3	24,4	27,9	34,8	29,1	20,4	14,5
1 white 2 black	21,0	10,9	9,1	9,3	7,8	4,6	5,2	4,2	7,3	6,6	5,8	3,2	2,5
3 black	1,6	0,8	0,4	0,7	0,7	0,7	0,3	0,5	0,7	0,5	0,6	0,4	0,4

4 Conclusions and Future Work

The motif analysis enables detection of the communication patterns within complex social networks. The approach presented in this paper extends existing motif analysis techniques by taking into account the properties (centrality) of network nodes. This is a continuation of results presented in [5], where the influence on the communication intensity on the local network topology was investigated. The obtained results revealed unknown properties of local topology of social networks, like differences between the role of DC and DP hubs, invisible for simple TSP analysis. They also form a basis for entirely new set of large dynamic social network analysis methods which build on several unique properties of motif analysis:

- Promising computational cost – even for huge networks TSPs can be obtained with fast sampling algorithms which offers possibility of frequent tracking of social network evolution processes.
- Possible integration with soft computing methods – colouring approach which attributes network nodes and edges with the values of arbitrary parameters coming from social network theory allows association of fuzzy variables (like “average DP”) with local network structures/nodes. Future network mining methods may build on this.

- The possibility of developing algorithms for inferring global properties (like clustering coefficient) of the network from the TSP – first promising results were presented in [3] for simple Watts-Strogatz social network structures.

The next research steps will include motif analysis with respect to more sophisticated parameters of social network nodes (like social position and betweenes centrality). Also the dynamics of the network will be addressed through: a) the influence of local topology changes on the network; b) the consequences of attaching/deleting nodes and edges; and c) the analysis of changes in the WUT social network during academic year – in order to check how known periodic changes in University's business profile and users' activity affect the local topology of its social network.

References

1. Alexander, C.N.: A method for processing sociometric data. Sociometry 26, 268–269 (1963)
2. Barabasi, A.-L., Albert, R.: Emergence of scaling in random networks. Science 286, 509–512 (1999)
3. Chung-Yuan, H., Chuen-Tsai, S., Chia-Ying, C., Ji-Lung, H.: Bridge and brick motifs in complex networks. Physica A 377, 340–350 (2007)
4. Itzkovitz, S., Milo, R., Kashtan, N., Ziv, G., Alon, U.: Subgraphs in random networks. Phys. Rev. E. 68, 26127 (2003)
5. Juszczyszyn, K., Musiał, K., Kazienko, P.: Local Topology of Social Network Based on Motif Analysis. In: 11th International Conference on Knowledge-Based Intelligent Information & Engineering Systems. LNCS (LNAI). Springer, Heidelberg (2008)
6. Kashtan, N., Itzkovitz, S., Milo, R., Alon, U.: Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. Bioinformatics 20(11), 1746–1758 (2004)
7. Mangan, S., Alon, U.: Structure and function of the feedforward loop network motif. Proc. Natl Acad. Sci., USA 100, 11980–11985 (2003)
8. Mangan, S., Zaslaver, A., Alon, U.: The coherent feedforward loop serves as a sign-sensitive delay element in transcription networks. J. Mol. Biol. 334, 197–204 (2003)
9. Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., Shen-Orr, S., Ayzenshtat, I., Sheffer, M., Alon, U.: Superfamilies of evolved and designed networks. Science 303(5663), 1538–1542 (2004)
10. Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., Alon, U.: Network motifs: simple building blocks of complex networks. Science 298, 824–827 (2002)
11. Proctor, C.H., Loomis, C.P.: Analysis of sociometric data. In: Jahoda, M., Deutch, M., Cok, S. (eds.) Research Methods in Social Relations, pp. 561–586. Dryden Press, NewYork (1951)
12. Shaw, M.E.: Group structure and the behavior of individuals in small groups. Journal of Psychology 38, 139–149 (1954)
13. Shen-Orr, S., Milo, R., Mangan, S., Alon, U.: Network motifs in the transciptional regualtion network of Escherichia coli. Nat. Genet. 31, 64–68 (2002)
14. Vazquez, A., Dobrin, R., Sergi, D., Eckmann, J.-P., Oltvai, Z.N., Barabasi, A.: The topological relationship between the large-scale attributes and local interaction patterns of complex networks. Proc. Natl Acad. Sci. USA 101, 17, 940 (2004)
15. Wasserman, S., Faust, K.: Social network analysis: Methods and applications. Cambridge University Press, New York (1994)
16. Young-Ho, E., Soojin, L., Hawoong, J.: Exploring local structural organization of metabolic networks using subgraph patterns. Journal of Theoretical Biology 241, 823–829 (2006)

Initialization Dependence of Clustering Algorithms

Wim De Mulder[1], Stefan Schliebs[2], René Boel[1], and Martin Kuiper[3]

[1] SYSTeMS, Ghent University, Technologiepark 914, 9052 Ghent, Belgium
{wim.demulder,rene.boel}@ugent.be
[2] Knowledge Engineering and Discovery Research Institute, Auckland University of Technology, 350 Queen Street, 1010 Auckland, New Zealand
sschlieb@aut.ac.nz
[3] Department of Biology, Norwegian University of Science and Technology, Høgskoleringen 5, 7491 Trondheim, Norway
kuiper@nt.ntnu.no

Abstract. It is well known that the clusters produced by a clustering algorithm depend on the chosen initial centers. In this paper we present a measure for the degree to which a given clustering algorithm depends on the choice of initial centers, for a given data set. This measure is calculated for four well-known offline clustering algorithms (k-means Forgy, k-means Hartigan, k-means Lloyd and fuzzy c-means), for five benchmark data sets. The measure is also calculated for ECM, an online algorithm that does not require the number of initial centers as input, but for which the resulting clusters can depend on the order that the input arrives. Our main finding is that this initialization dependence measure can also be used to determine the optimal number of clusters.

1 Introduction

Clustering is the partitioning of a data set into subsets (clusters), so that the data elements in each subset are similar to each other and dissimilar to the data elements in other subsets. Similarity and dissimilarity are defined in terms of a distance measure.

Many clustering algorithms start with initial centers and repeat a given rule to transform these centers to representatives of clusters. Ideally the clusters that are recognized in a given data set depend only on the structure of the data set and the chosen distance measure. However it is known that the choice of initial centers influences the resulting clusters and consequently research concentrates on decreasing the degree of dependence on initial centers [1,2,3,4]. However little research has taken place that actually explores the degree to which clustering algorithms are initialization dependent. This is probably caused by the lack of a suitable measure for initialization dependence (hereafter shortened as ID). In this paper we present such a measure. We start from the idea that a clustering algorithm itself acts as a deterministic function between random initializations of the algorithm (*e.g.* initial centers, order of data input) and the resulting set of

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 615–622, 2009.

cluster centers. A resulting set of clusters, hereafter called a clustering, can thus be viewed as a generalization of a random variable. In section 2 we describe this statistical view in more rigorous terms and extend the basic statistical notion of the dispersion of a random variable, the variance, to a measure for the dispersion of a random clustering. In section 3 we present experimental results and notice that the ID measure reaches minimal values when the number of clusters equals the optimal number. This suggests the use of the ID measure as an optimization tool for the number of clusters, which is explored in section 4. In section 5 we conclude our findings.

2 Material and Methods

2.1 Initialization Dependence Measure

Suppose a data set $D = \{a_1, \ldots, a_n\}$ that is to be clustered by a given hard clustering algorithm, i.e. an element either belongs to a cluster or does not belong to it. A clustering produced by this algorithm can be represented by a matrix C with elements $C(i,j), i,j = 1, \ldots, n$, where $C(i,j) = 1$ if a_i and a_j belong to the same cluster, and $C(i,j) = 0$ if a_i and a_j belong to different clusters. We call such a matrix a cluster matrix. Obviously there is a unique correspondence between a clustering and a cluster matrix which implies that the terms clustering and cluster matrix can be used interchangeably.

The elements $C(i,j)$ can be interpreted as the random outcomes associated with random variables $A(i,j)$, their distribution determined by the given clustering algorithm. The matrix A with elements $A(i,j)$ is then called a random clustering. The notation C is used for a random outcome clustering, i.e. a matrix that contains outcomes of the random variables in A. One execution of a clustering algorithm, yielding one random outcome clustering, is called a run.

We use the notations $p_{ij}(0)$ and $p_{ij}(1)$ to denote the probability that $A(i,j)$ is 0 and $A(i,j)$ is 1 respectively. The expected value of $A(i,j), \mathrm{E}[A(i,j)]$, is thus $p_{ij}(0) \times 0 + p_{ij}(1) \times 1 = p_{ij}(1)$. The variance of $A(i,j)$, $\mathrm{Var}(A(i,j))$, is given by $E[(A(i,j) - E[A(i,j)])^2]$. This simple statistical framework allows us to extend the expected value of a random variable to the expected value of a random clustering. We simply define the expected clustering $\mathrm{E}[A]$ as the matrix with element $\mathrm{E}[A(i,j)]$ on position (i,j). The variance of a random clustering A, $\mathrm{Var}(A)$, is defined as the matrix with elements $\mathrm{Var}(A(i,j))$. This matrix provides information about the dispersion of a random clustering. A matrix norm is useful to reduce the information in this matrix to one real value. We thus define the ID of a random clustering A as $||\mathrm{Var}(A)||$ for some matrix norm $||.||$. We will use

$$||\mathrm{Var}(A)|| = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{n} \mathrm{Var}(A(i,j)) \tag{1}$$

The motivation for dividing by $n(n-1)$ rather than by n^2 is that the elements on the diagonal are always 1, and thus contain no relevant information.

2.2 Clustering Algorithms and Parameters

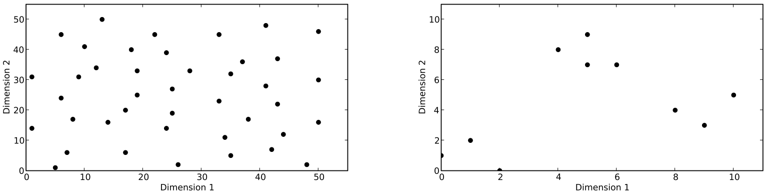
In section 3 we apply the ID measure (1) to five clustering algorithms: k-means [5] with the implementations of Forgy, Hartigan and Lloyd, the fuzzy algorithm c-means [5] and ECM [6]. K-means is a hard clustering algorithm for which the cluster matrix representation of section 2.1 is suitable. However fuzzy c-means is a soft algorithm: a set of clusters is produced and each element is given a membership value between 0 and 1 in each cluster. An explicit relationship between two given elements is not established and the matrix representation is therefore not directly applicable. To solve this representation problem we construct the matrix C in an alternative way. The c-means algorithm gives as output $\lambda(a_i, K_l)$, the membership value of a_i in a cluster K_l, for all data elements a_i and all clusters K_l produced by the algorithm. We convert the degrees to which elements belong to each cluster, $\lambda(a_i, K_l)$, to degrees to which two given elements belong together (which will be used as the needed values $C(i,j)$), as follows. Suppose that $\lambda(a_i, K_l) = \lambda(a_j, K_l) = 1$ and $\lambda(a_i, K_p) = \lambda(a_j, K_p) = 0$ for all $p \neq l$. This corresponds with the hard clustering case where a_i and a_j belong to the same cluster, i.e. $C(i,j) = 1$. Suppose now that $\lambda(a_i, K_l) = 1$, $\lambda(a_i, K_p) = 0$ for all $p \neq l$, $\lambda(a_j, K_r) = 1$ with $r \neq l$ and $\lambda(a_j, K_t) = 0$ for all $t \neq r$. This corresponds with the hard clustering case where a_i and a_j belong to different clusters, i.e. $C(i,j) = 0$. We notice that in both cases $C(i,j)$ corresponds with the maximum of the set $\{\min(\lambda(a_i, K_p), \lambda(a_j, K_p))\}$ over all clusters K_p produced by the soft algorithm. We use this observation as definition for $C(i,j)$:

$$C(i,j) = \max\{\min_{K_p \text{ is a cluster}} \{\lambda(a_i, K_p), \lambda(a_j, K_p)\}\} \tag{2}$$

ECM is an online algorithm that produces hard clusters and so the matrix representation is suitable. Euclidean distance was used as distance metric for all algorithms. The initial centers were chosen to range from 2 to 14. We make an exception for spaeth2_04, because it only consists of 10 data points. The number of initial centers for this data set was chosen to range from 2 to 5.

ECM does not require to predefine the number of initial centers, but it does require to set a radius parameter r. This parameter determines when a new cluster has to be created for an incoming data point. This parameter was chosen to range from 0.025 to 0.5 in steps of 0.025. Because of the evolving nature of ECM the input order of the given dataset is relevant and impacts the cluster generation. In our experiments the data order was varied in each run of ECM.

In theory we have to calculate $\text{Var}(A)$ for a random clustering A to obtain the ID measure (see section 2.1). However because the distribution of the random variables in this matrix is not known, we have to rely on approximation techniques. Therefore a sample S of N clusterings $S = \{C_1, \ldots, C_N\}$ is generated and we calculate an approximation for the ID measure, $\hat{ID}$, using this sample. If we use the notation $\bar{C}$ for the average cluster matrix consisting of elements $\bar{C}(i,j) = 1/N \sum_{k=1}^{N} C_k(i,j)$, we define $\hat{ID} = ||\hat{\text{Var}}(A)||$. $\hat{\text{Var}}(A)$ is a matrix with element $1/(N-1) \sum_{k=1}^{N} (\bar{C}(i,j) - C_k(i,j))^2$ on position (i,j). The matrix norm



(a) The spaeth_02 data set

(b) The spaeth_04 data set

Fig. 1. Data sets from the SPAETH library. In a) no clear clusters can be identified, while b) seems to contain three clusters.

$||.||$ is given by (1). Just as $\mathrm{Var}(A)$ is an extension of the variance of a random variable, $\hat{\mathrm{Var}}(A)$ is an extension of the unbiased estimator of the population variance. The parameter N was chosen as 400.

2.3 Data

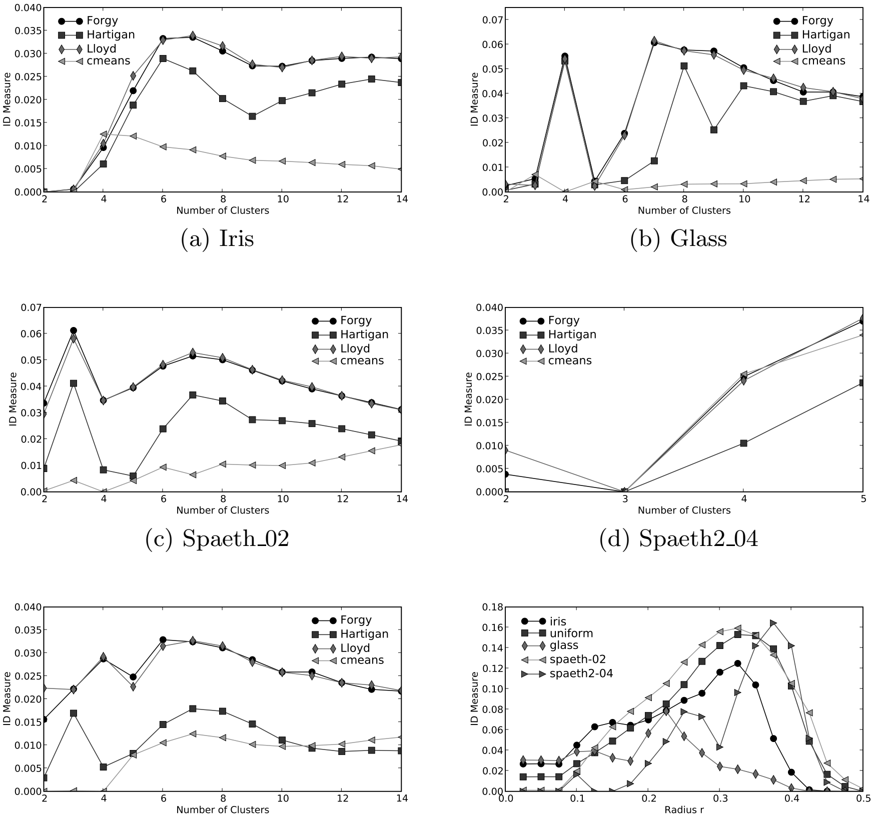
Five data sets were used for the experimental results in section 2.2. Two of these data sets are taken from the UCI Machine Learning Repository [7] which contains real life data sets. The first is the iris data set which is ideally clustered in 3 groups. The second is the glass identification data set which contains 7 different glass types. However if the data set is viewed at a higher level it only contains 5 classes, because we can expect that the differences between "windows for buildings - float processed" and "windows for buildings - non float processed" are minor, and also for "windows for vehicle - float processed" and "windows for vehicle - non float processed". Two other data sets are spaeth_02 and spaeth2_04 taken from the SPAETH [8] and SPAETH2 [9] library respectively, which contains artificial data sets to test clustering algorithms. Each data item in these sets contains two numerical attributes which allows plotting the data, see Fig. 1. It seems that spaeth2_04 contains 3 clusters, but that in spaeth_02 no reasonable clusters can be detected. We also generated a random uniform data set, consisting of 50 instances with two attributes. Each attribute was uniformly generated between 0 and 10.

3 Experimental Results

We applied the five clustering algorithms (see section 2.2) to the five data sets (see section 2.3), where the parameters were set to the values mentioned in section 2.2 and calculated $\hat{ID}$. The results are shown in figure 2.

The X-axis contains the number of clusters, except for ECM. The results for ECM have been put in one figure, with the X-axis containing the radius.

We first concentrate on the offline algorithms. It is clear that Forgy and Lloyd give very similar values for the ID measure, whereas the Hartigan implementation is less sensitive for initial values. However the fuzzy algorithm c-means is for



(a) Iris

(b) Glass

(c) Spaeth_02

(d) Spaeth2_04

(e) Uniform

(f) ECM

Fig. 2. Evolution of the ID measure for the tested data sets. In (a)-(e) a data point corresponds to the ID measure of a given clustering method for a defined cluster number. (f) presents the results for ECM, here each trajectory corresponds to a data set.

most data sets and for most number of clusters considerably less initialization dependent. This suggests that soft clustering algorithms have the advantage of being less sensitive for the choice of initial centers than hard clustering algorithms.

Consider now the ID measure when the number of clusters equals the optimal number. For iris this optimum is 3 (see section 2.3), and Fig 2a shows that the ID is also minimal there for all clustering algorithms. For glass the correct number of classes is 5 or 7 depending on the level we view the data set. The algorithms agree that for 5 clusters the sensitivity for initial centers is very low. The difference between "float processed" and "non float processed" is apparently too small to be noticed by any of the algorithms. Spaeth_02 is a data set for which it is difficult to recognize clusters, see Fig 1a. However Hartigan and c-means are able to group the data well in 2, 4 and 5 clusters. Forgy and Lloyd are not able to cluster this data set with a small sensitivity for initial centers. The values for their ID measure exceed 0.03. This is large compared to their minimum values (below 0.01) for iris, glass and spaeth2_04. spaeth2_04 is a data set that suggests 3 clusters, see Fig 1b. All algorithms agree that this number of clusters gives a minimal dependence on initial centers. For the random, uniform data set it is expected that no reasonable clusters can be found. Again, Hartigan and c-means are able to recognize groups in this data set with little sensitivity to initial centers. The behavior of Forgy and Lloyd is to be compared with their behavior for spaeth_02: for all numbers of initial centers the ID is high. This analysis suggests that the ID measure can be used to determine the optimal number of clusters. We explore this idea in section 4.

ECM generally reports a higher ID measure on all data sets, which is due the fact that the number of evolved clusters can vary for a specific parameter r. This is quite in contrast to the other clustering methods, where the exact cluster number is specified beforehand. For small values ($r \leq 0.075$) many samples are likely to be clustered into individual clusters, which results in a small ID measure. On the other hand it seems that for large r all samples belong to one single cluster, which is also reflected by a minimal ID measure. The local minima in between these trivial solutions are expected to represent the optimal parameter configuration for ECM. This is in particular true for the spaeth2_04 data set: Setting $r = 0.125$ and $r = 0.15$ results on average in the evolution of exactly three clusters in each of the 400 independent runs. As expected ECM can not identify any cluster in the uniform data set. The spaeth_02 data has apparently very similar characteristics as the uniform data, since both ID measure profiles show similar trajectories and both lack an additional local minimum. For the iris and the glass data sets the situation is less clear. Both ID measure profiles have local minima at $r = 0.175$. We have determined how many clusters have been on average obtained by ECM when choosing $r = 0.175$: For the glass data this setting corresponds to approximately 11.08 clusters (0.810), while for the iris data on average 5.6 (0.819) cluster were evolved (values in brackets present the standard deviation). A closer analysis of the obtained clusters showed that ECM could distinguish between the expected clusters, but additionally evolved several sub-clusters within these expected clusters. The sub-clusters displayed large variations between independent runs, which results in an increased ID measure for larger radius values $0.2 \leq r \leq 0.35$. Choosing a suitable radius seems a non-trivial task, especially for data sets containing overlapping clusters, as in the case for the iris and glass data.

4 Use of ID Measure to Find Optimal Number oF Clusters

The analysis in section 3 for the offline algorithms suggests that the initialization dependence of clustering algorithms is small (if not minimal) when the number of initial centers equals the optimal number of clusters. Furthermore, if the data set can not be clustered in a reasonable way (in this case spaeth_02 and the uniform data set), the ID of several clustering algorithms is high for all numbers of initial centers. This was especially true for Forgy and Lloyd.

It is traditional practice that cluster validity measures, e.g. the Davies-Bouldin index [10], are used to determine the optimal number of clusters. An important disadvantage of these cluster validity measures is that they favor certain special forms of clusters, especially 'circle shaped' clusters. More general forms of clusters typically result in poor values for these validity measures (even if the separation between the clusters is high). This disadvantage is due to the fact that these validity measures rely on a given distance metric to measure the within-cluster scatter and the between-cluster separation. The use of a distance measure presupposes that we have an idea of how the data is distributed, because the distance measure determines the meaning of similarity between elements and thus the meaning of a qualitative cluster.

Our above analysis proposes that the ID measure can be used as an alternative tool to determine the optimal number of clusters, because the ID is low for the optimal number of clusters. This measure has the advantage that it is not restricted to certain data distributions. Whatever the distribution is, if the data set can be optimally clustered in m clusters and the clustering algorithm is able to recognize these m clusters, it is expected that the good separation of the clusters ensures a large insensitivity for initial centers. This will then be revealed by the low value of the ID measure for m clusters. Of course, if the algorithm is only able to recognize a few typical data distributions, the ID measure will also not be able to give the optimal number of clusters if the data set has another distribution, but this deficiency is due to the algorithm and not to the ID measure. Furthermore we can get round this deficiency by plotting the ID measure for several clustering algorithms (possibly with different distance measure).

Another advantage of the ID measure is that it is able to indicate that a given data set can not be clustered by means of a given distance measure. This is clear when the values for $\hat{ID}$ for spaeth_02 and the uniform data set are compared with those for iris, glass and spaeth2_04. For the ECM it was found that the bevahior of the ID for the uniform and spaeth_02 data sets (linear increase-linear decrease) was rather different from the bevahior of the other three data sets.

However if the ID measure is used to determine a good number of clusters, we should keep in mind that the figures suggest to calculate the ID measure for several algorithms. The combination of the several plots gives a rather accurate view on the optimal number of clusters whereas this is not necessarily valid if we restrict us to one clustering algorithm. For example c-means indicates that the glass data set can be clustered in any number of clusters (ranging from 2 to 14), and k-means Lloyd suggests that the uniform data set can equally well be clustered in 2, 3 or 5 clusters. If however all algorithms have a fairly low value for a given data set for a certain number of clusters, we can be rather sure that this number of clusters is optimal (or near-optimal) for the data set.

The analysis in section 3 also reveals an unexpected advantage of algorithms that are rather sensitive to initial centers, like Forgy and Lloyd. Fig. 2 and the analysis in section 3 suggest that Forgy and Lloyd are better indications of the inability of the algorithm to cluster a given data set. For these algorithms the values of the ID measure for spaeth_02 and the uniform data set are considerably higher than for the other data sets, and this is true for all considered numbers of initial centers. We can thus not state that algorithms with a high ID are inferior to those with a low ID. Section 3 suggests that they are complementary and that their combination can give a reliable answer to two questions: a) Can the given data set be clustered in a satisfactory way, and b) if the data set can be clustered, what is the optimal number of clusters?

5 Conclusion

In section 2.1 a measure for the dependence of clustering algorithms on initial centers was presented. This measure was applied to five data sets for five clustering algorithms in section 3. It was experimentally found that the measure has minimal or near-minimal values when the number of clusters equals the optimal number, which suggests the use of this measure to find the optimal number of clusters. This idea was explored in section 4. Further research is needed to confirm this experimental observation. Two advantages of this initialization dependence measure over classical cluster validity measures in selecting the optimal number of clusters are a) the developed measure is distribution-free and b) it is able to indicate that a given data set can not be clustered in a satisfactory way.

Acknowledgments. Financial support from BOF (Special Research Fund) Ghent University is gratefully acknowledged.

References

1. Redmond, S.J., Heneghan, C.: A method for initialising the K-means clustering algorithm using kd-trees. Pattern Recognition Letters 28, 965–973 (2007)
2. Al-Daoud, M.B., Roberts, S.A.: New methods for the initialisation of clusters. Pattern Recognition Letters 17, 451–455 (1996)
3. Katsavounidis, I., Kuo, J., Zhen Zhang, C.-C.: A new initialization technique for generalized Lloyd iteration. IEEE Signal Processing Letters 1, 144–146 (1994)
4. Khan, S.S., Ahmad, A.: Cluster center initialization algorithm for K-means clustering. Pattern Recognition Letters 25, 1293–1302 (2004)
5. Jain, A.K., Murty, M.N., Flynn, P.J.: Data Clustering: A Review. ACM Computing Surveys 31, 264–323 (1999)
6. Kasabov, N.: Evolving Connectionist Systems: The Knowledge Engineering Approach. Springer, Heidelberg (2007)
7. UC Machine Learning Repository, `http://archive.ics.uci.edu/ml/`
8. SPAETH Cluster Analysis Datasets, `http://people.scs.fsu.edu/~burkardt/datasets/spaeth/spaeth.html`
9. SPAETH2 Cluster Analysis Datasets, `http://people.scs.fsu.edu/~burkardt/datasets/spaeth2/spaeth2.html`
10. Davies, D.L., Bouldin, D.W.: A Cluster Separation Measure. IEEE Transactions on Pattern Analysis and Machine Intelligence 1, 224–227 (2000)

Boundary Detection from Spectral Information

Jun Ma

School of Information Security Engineering,
Shanghai Jiao Tong University, Shanghai 200240, China
mjnicky@gmail.com

Abstract. In this paper, we propose a method to detect object boundaries from spectral information. Previous image boundary detection techniques draw their attention on spatial image features such as brightness, color and texture. Different from traditional feature descriptor methods, we started from the analysis of natural image statistics in spectral domain and proposed a method to detect image boundaries by analyzing log spectrum residual of images. We find that the spatial transform of log spectrum residual of images are qualified as boundary maps. In the experiment section we show that our results are similar to human segmentations compared to common methods like the Canny detector.

1 Introduction

Boundary detection has been an active research topic in computer vision. In general, boundary is referred to a contour in the image that represents a change of pixel ownership from one object to another. Boundary detection is a low-level task in both human vision systems and computer vision applications. It is substantial to some mid-level and high-level tasks such as image segmentation and object detection. In this paper, we propose a new approach to detect object boundaries without concerning how to use this model in further mid-level and high-level applications.

The most commonly used boundary detection algorithms focus on the discontinuity in image brightness, such as the Canny detector [1]. These algorithms are inadequate models since they fail to recognize the texture patterns in the image and fire wildly inside these regions. To improve the performance, some other image features are computed, such as color, texture or even some high-level cues like shape. Texture analyze is the most discussed one among them. Various texture descriptors have been introduced, such as [2, 3]. However, while they work well on pure texture-texture boundaries in the images, some simple brightness change are neglected due to the limited information contained in mere texture. In [4], the author proposed a framework for boundary detection which combines many image features: local brightness, color and texture cues. The cue combination is modeled as a supervised learning problem, enables the algorithm to learn the cue combination rules from the human segmentation as ground truth data provided by the Berkeley Segmentation Dataset [5]. The authors also propose a precision-recall benchmark algorithm to evaluate their boundary detection framework, which we also adopt in this paper. However, their method

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 623–630, 2009.

relies on prior knowledge about the feature combination (human segmentation as ground-truth data for machine learning), and the complicated framework is difficult to implement.

In this paper, a boundary detection method which conducts spectral information is proposed. In Section 2, the model of spectral residual is introduced, which is the model we used in this paper to generate image boundaries. We start from analyzing natural image statistics and their log spectrum features, and then we obtain the spectral residual in the frequency domain and transform it into spatial domain to obtain boundaries in the image. In Section 3, we introduce the benchmark method we adopt from [4] and provide the experiment result of our algorithm.

2 Image Boundaries from Spectral Residual

2.1 The Log Spectrum Representation

One invariant feature of natural image statistics that has been widely studied is scale invariance [6, 7], which is known as 1/f law. It concludes that the average Fourier transform of a large group of images obey the distribution:

$$E\{A(f)\} \propto 1/f \tag{1}$$

This feature can be easily recognized from the log-log spectrum of images (See Fig.1.(5)). However, this linear feature in log-log spectrum cannot be used to analyze single image, since the log-log spectrum pattern of a single image is not a straight line in most cases (Fig.1). Moreover, the sampling points are not well proportioned in the log-log spectrum: the high-frequency parts draw together, which leads to noise [8]. To overcome this defect, the author in [9] proposed the model of log spectrum, the comparison of log spectrum and log-log spectrum is shown in Fig.1. By analyzing the log spectrum of natural images, the author in [9] finds that the spectral residual in log spectrum domain has some special features in its spatial transform. The spectral residual is defined as following, given an input image $Im(x)$, we have:

$$A(f) = \Re(F[Im(x)]) \tag{2}$$

As the amplitude (real) part of $Im(x)$'s fourier transform and:

$$P(f) = \Im(F[Im(x)]) \tag{3}$$

As the Phase (Imaginary) part of the $Im(x)$'s fourier transform.

$$L(f) = log(A(f)) \tag{4}$$

$$R(f) = L(f) - Avglog(f) \tag{5}$$

$$Rsd(x) = F^{-1}[exp(R(f) + P(f))] \tag{6}$$

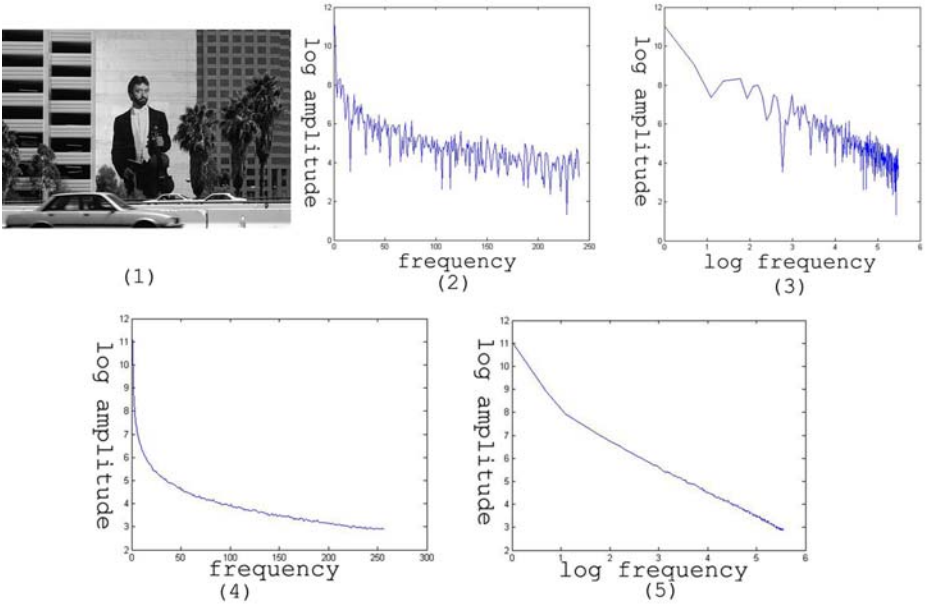


Fig. 1. log spectrum and log-log spectrum representation. (1)single image (2)log spectrum of the single image (3)log-log spectrum of the single image (4)log spectrum of 300 images (5)log-log spectrum of 300 images.

For each input image $Im(x)$, we compute the amplitude and phase of its Fourier Transform, $A(f)$ and $P(f)$, then we obtain the log spectrum residual by subtracting the average log spectrum $Avglog(f)$ from the input image's log spectrum $log(A(f))$. After that, an inverse Fourier Transform is conducted to get the spatial image of the spectral residual. This procedure is illustrated in Fig 2.

2.2 Spectral Residual to Object Contour

The authors in [9] used this special feature of the spectral residual map to generate saliency map as a computational model of visual attention. From Fig. 2 we can clearly see that the bright pixel clusters are located definitely at the proto-objects in the images. After a large amount of experiments on the spectral residual model, we find that the spectral residual map can be used to generate more meaningful maps - the boundary maps, which points out the objects in the image more accurately by figuring out their contours rather than point out proto-object locations. However, the raw result of spectral residual cannot be directly used as the result of boundary maps because it is only a group of discreet and randomly distributed bright pixels lies near the object boundaries in the image (See Fig.2). To make the result more like continuous boundary contours, we should smooth these bright pixels into lines and curves, and we find the Gestalt psychology quite applicable here.

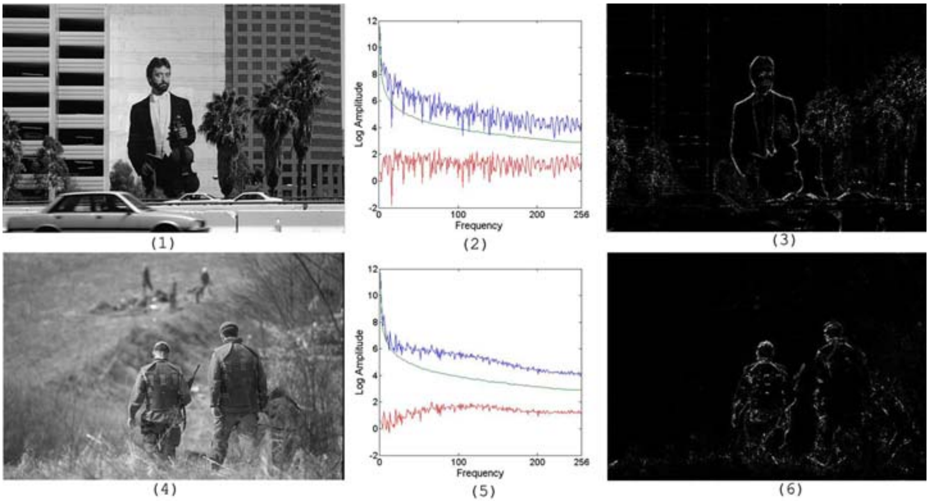


Fig. 2. Spatial feature of log spectrum residual. Left column: The input image. Middle column: The log spectrum of input image(blue), the average log spectrum(green), the log spectrum residual(red). Right column: Inverse Fourier Transform of the spectral residual into spatial images with boundary features.

The Gestalt psychology starts from the research of visual grouping in human vision systems [10] and is concluded into a series of laws that can predict the human vision behavior, called the Gestalt laws. One of these laws, the law of continuity points out that human beings tend to recognize continuous contours from the visual input, and this law has become the guiding rule of many computer vision applications [11]. Figure 3 shows one hundred 13 × 13 patches, which are the most common image boundary patches computed from 300 human segmented images, the Berkeley Segmentation Database [12]. These 100 patches account for 74.78% boundary patches in more than 10,000 patches we obtained from the database. We can see clearly that most of the boundary patches are direct lines with different orientations. Based on this statistics, we have designed a group of linear filters to smooth the raw result of spectral residual image, as shown in Fig.4. These filters are similar to line detectors in ordinary image processing techniques, such as Gabor filters. In this task, they are quite qualified to smooth the discreet bright pixels in our raw spectral residual map into continuous lines. The procedure of smoothing is illustrated as blow:

The spectral residual map $Rsd(x)$ is convoluted with each line detector patch $P(x)_i$ to obtain several direction maps $D(x)_i$:

$$D(x)_i = P(x)_i * Rsd(x) \tag{7}$$

The orientation map $O(x)$ is defined as the max value among different direction maps:

$$O(x) = \max_i D(x)_i \tag{8}$$

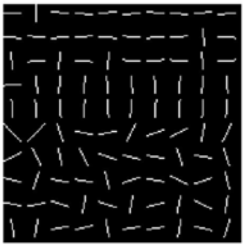


Fig. 3. The 100 most frequent image boundary patches computed from 300 natural images in [12]

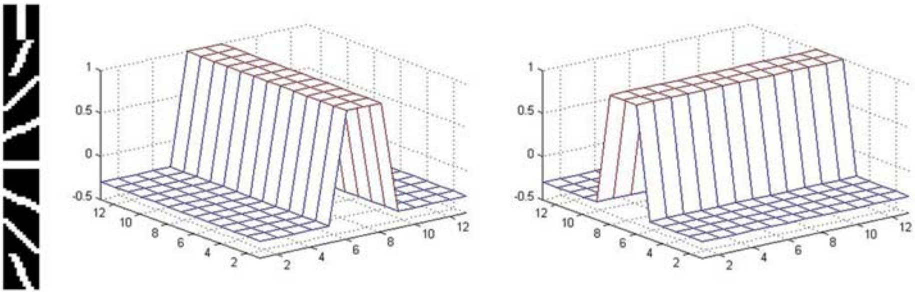


Fig. 4. The filters to smooth the raw image results

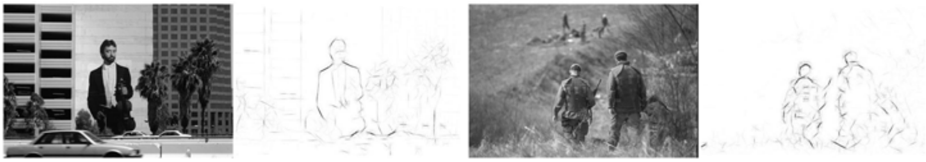


Fig. 5. The result after smoothing

Moreover, the number i of $D(x)_i$ that has the max convolution value is defined as the direction of each pixel, named as the orientation label map $OL(x)$:

$$OL(x) = arg \max_i D(x)_i \tag{9}$$

The final boundary map $B(x)$ is the product of orientation map $O(x)$ and length map $L(x)$:

$$B(x) = O(x) \times L(x) \tag{10}$$

The length map $L(x)$ is defined as the number of same direction pixels in the 13×13 patch centered at the position x in the orientation label map $OL(x)$.

The result of our boundary detection algorithm after smoothing is illustrated in Fig 5.

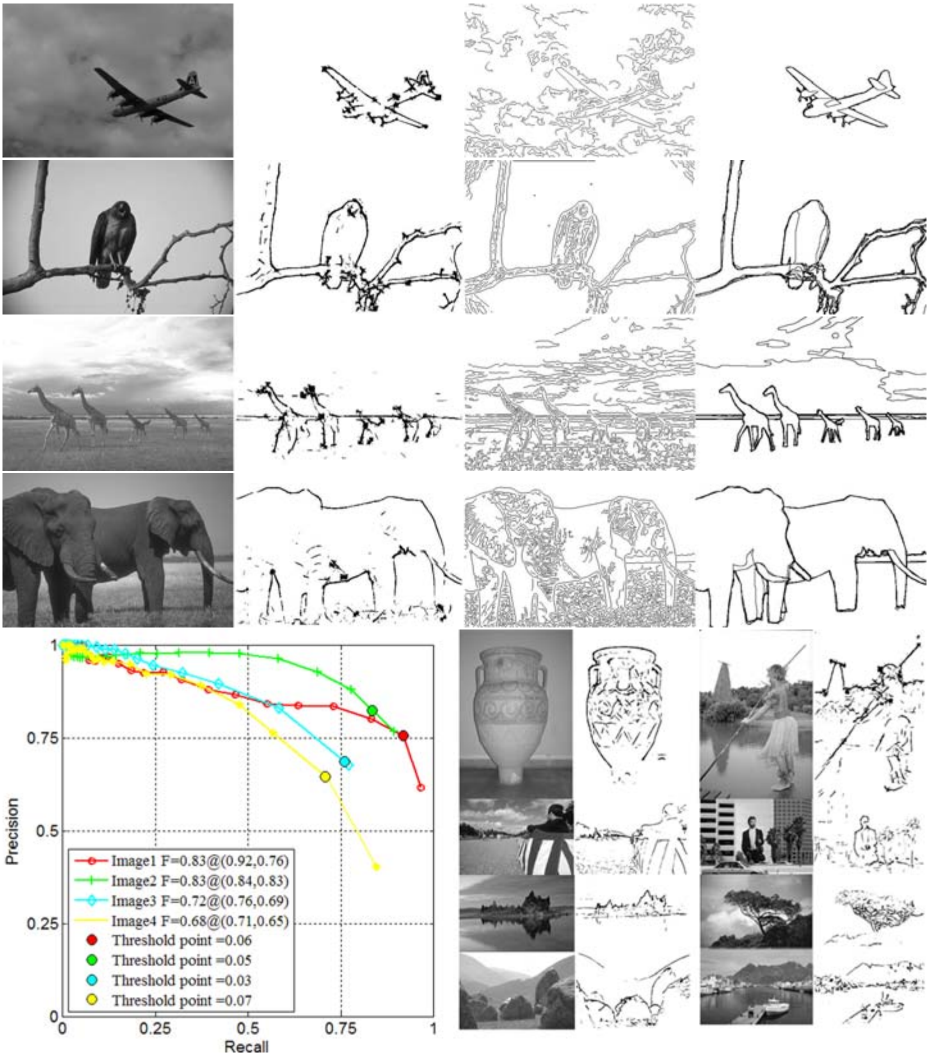


Fig. 6. The result of our boundary detection algorithm (upper Column 2) compared to MATLAB Canny detector with automatic parameters (upper Column 3) and human segmentations (upper Column 4) and the precision-recall curves of these images (bottom-left). More results with our boundary detection method(bottom-right)

3 Experiment and Benchmark

In [4], the authors proposed a precision-recall framework to evaluate different boundary detection algorithms using human labeled segmentation as ground truth data. In their framework, the precision denotes the fraction of detected boundaries that are true positives and the recall denotes the fraction of true boundaries that have been detected. The precision-recall curve is a curve connecting a group of points with different thresholds applied on the final boundary map. Fig 6 illustrates the result of our boundary detection algorithm with its comparison to the common Canny detectors and human segmentations, all the images are from the Berkeley Segmentation Database. We can see that our results are close to human segmentation results. And in the second part of Fig. 6 we provide the precision-recall curve of each image above. From the experiment results and the benchmark curves, we conclude that our method provides high precisions while it is difficult for our boundary map to reach high recall area in the precision-recall curve, this phenomenon is mainly because the unbalanced proportion of brightness distribution introduced by the spectral residual model: only a small amount of pixels have very high brightness values while most pixels in the image have a value close to zero. This special feature also makes sense in the threshold determination issue of our method: The boundaries and non-boundaries are naturally divided due to the discontinuous brightness distribution. To better illustrate this feature, we compute the corresponding F-measure values of our precision-recall curves [4], which is very helpful for us to determine the optimal threshold from existing P-R curves. The F-measure is defined as

$$F = PR/(\alpha R + (1 - \alpha)P) \tag{11}$$

It captures the trade-off between Precision and Recall by a relative cost α. The point along the precision-recall curve which has the greatest F-measure values provides us the optimal detector threshold for our boundary map. The detection results shown in Fig 6 are all conducted by this optimal threshold. Note that in this paper, we set our α to 0.5, which means that precision and recall are equally important in the determination of thresholds. From the result, we find that among the 300 images in the database, the optimal threshold for images are ranged in a relatively small area(0.03 - 0.1). This special feature enables us to define a global threshold without the need to conduct some learning procedures while preserving the quality of the result at the same time.

4 Discussions

In this paper, we proposed a new method to detect image boundaries which conduct information in frequency domain rather than spatial domain. The result of our experiment shows that our boundary maps are close to human labeled boundaries. However, there are still some limitations in our algorithm. First, the algorithm don't work well on random textures with large brightness changes. For example, see the trees in the left image of Fig.5, we can see that the method fire falsely inside these areas. To overcome this defect, we should integrate more cues into the boundary detection framework, such as texture cues. We believe that the combination of information from spatial and frequency domain can lead to better performance in this task. Moreover, it is difficult to make use of our low level boundary map in mid-level and high-level computer vision tasks, such as image segmentation and object recognition. This is what we will emphasis in our future research work.

References

1. Canny, J.: A Computational Approach to Edge Detection. IEEE Trans. Pattern Analysis and Machine Intelligence 8, 679–698 (1986)
2. Rubner, Y., Tomasi, C.: Coalescing Texture Descriptors. In: Proc. ARPA Image Understanding Workshop (1996)
3. Will, S., Hermes, L., Buhmann, J.M., Puzicha, J.: On Learning Texture Edge Detectors. In: Proc. Int'l Conf. Image Processing, pp. 877–880 (2000)
4. Martin, D., Fowlkes, C., Malik, J.: Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues. IEEE Trans. Pattern Analysis and Machine Intelligence. 26(1) (January 2004)
5. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In: International Conference on Computer Vision 2001 (2001)
6. Ruderman, D.: The Statistics of Natural Images. Network: Computation in Neural Systems 5(4), 517–548 (1994)
7. Srivastava, A., Lee, A., Simoncelli, E., Zhu, S.: On Advances in Statistical Modeling of Natural Images. Journal of Mathematical Imaging and Vision 18(1), 17–33 (2003)
8. van der Schaaf, A., van Hateren, J.: Modelling the Power Spectra of Natural Images: Statistics and Information. Vision Research 36(17), 2759–2770 (1996)
9. Hou, X., Zhang, L.: Saliency Detection: A Spectral Residual Approach In: CVPR 2007 (2007)
10. Koffka, K.: Principles of Gestalt psychology. Hartcourt, Brace and World, New York (1935) (1967)
11. Desolneux, A., Moisan, L., Morel, J.-M.: Gestalt Theory and Computer Vision, Theory and Decision Library, vol. 38
12. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In: Proc. 8th Int'l Conf. Computer Vision 2001 (2001)
13. Berkeley Segmentation and Boundary Detection Benchmark andDataset (2003), http://www.cs.berkeley.edu/projects/vision/grouping/segbench

Improvement of Practical Recurrent Learning Method and Application to a Pattern Classification Task

Mohamad Faizal bin Samsudin and Katsunari Shibata

Department of Electrical and Electronic Engineering
Oita university, 700 Dannnoharu, Oita 870-1192 Japan
shibata@cc.oita-u.ac.jp

Abstract. Practical Recurrent Learning (PRL) has been proposed as a simple learning algorithm for recurrent neural networks[1][2]. This algorithm enables learning with practical order $O(n^2)$ of memory capacity and computational cost, which cannot be realized by conventional Back Propagation Through Time (BPTT) or Real Time Recurrent Learning (RTRL). It was shown in the previous work[1] that 3-bit parity problem could be learned successfully by PRL, but the learning performance was quite inferior to BPTT. In this paper, a simple calculation is introduced to prevent monotonous oscillations from being biased to the saturation range of the sigmoid function during learning. It is shown that the learning performance of the PRL method can be improved in the 3-bit parity problem. Finally, this improved PRL is applied to a scanned digit pattern classification task for which the results are inferior but comparable to the conventional BPTT.

1 Introduction

The significance of recurrent neural networks (RNNs) is expected to grow more and more hereafter for developing higher functions due to its ability of purposive learning to memorize information or events that have occurred earlier in a sequence. Currently, there are two popular learning algorithms for recurrent neural networks, BPTT[3] and RTRL[4], that have been widely used in many application areas. However, the critical drawback of the conventional algorithms is that they suffer from the necessity of large memory capacity and computational cost.

BPTT requires $O(n^2T)$, order of computational cost and $O(nT)$, order of memory capacity where n is the number of nodes and T is the number of steps for tracing back to the past. That means the past T states of the neural network have to be stored and the learning is done by using them. However, if T is small, it is worried that sufficient learning according to the past state cannot be done. On the other hand, RTRL needs as large as $O(n^3)$ for memory capacity, and $O(n^4)$ for computational cost, in order to modify each connection weight without tracing back to the past. Using RTRL in a large scale network is impractical

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 631–638, 2009.

because of the explosion in necessary memory capacity and computational cost. So far, many researches such as [5] have been done based on BPTT and RTRL.

S. Hochreiter and J. Schmidhuber[6] have proposed a special network architecture that has some memory cells that enables constant, non-vanishing error flow within the memory cell. They used a variant RTRL and only O(n2) of computational cost is required. However, special structure is necessary and it cannot be applied to the general recurrent neural networks.

Therefore, clearly, a practical learning algorithm for recurrent neural networks that need $O(n^2)$ or less memory and $O(n^2)$ or less computational cost is required where n^2 is equivalent to the number of synapses. PRL is an algorithm that capable to keep the order of memory size and computational cost as low as $O(n^2)$, by introducing some variable to hold some past states which enables constant memory and local computation to be assigned at each synapse. Therefore, this does not only reduce the memory capacity and computational cost drastically, but also increases the feasibility as a hardware system. In the previous work, it was shown that benchmark problems (sequential EXOR and 3-bit parity problem) could be learned successfully by PRL even though the learning performance was often quite inferior to the conventional BPTT.[1]

This paper presents an extension of the PRL method. The target is to make PRL perform equivalently to or outperform the conventional methods, considering that PRL already excels in computational cost and memory size. The extension is made by adjusting hidden nodes' output to prevent monotonous and biased oscillation during learning as will be described in the next section. Finally, we apply this extended PRL method to a more difficult task and compare it to BPTT.

This paper is organized as follows. In section 2, the extended of PRL method in the discrete time domain is introduced. Section 3 presents the improved result for 3-bit parity and the application to a pattern recognition task. Section 4 presents the conclusion of this paper.

2 Practical Recurrent Learning (PRL)

This section briefly recounts the PRL method in the discrete time domain as proposed in [1] at first. The forward calculation is the same as a regular neural network[3] in which each node computes weighted sum of its inputs and nonlinear transformation by sigmoid function. The basic idea is, some variables that are allocated to each synapse and hold the past information are introduced, considering the relationship between the outputs of post and pre-synaptic nodes. The connection weights between the nodes are modified by using the variables and propagated error signal. In order to keep the memory size and calculation time small, the error is propagated backwards like conventional BP[3] without tracing back to the past. In the past work[2], in the continuous time domain,

[1] Comparison to the RTRL is not shown in this paper, considering that RTRL is less practical than BPTT in terms of memory capacity and computational cost in larger networks.

three kinds of variables to hold the following items of the past information were introduced intuitively based on trials and errors.

1. the latest outputs of pre-synaptic nodes,
2. the outputs of pre-synaptic nodes that change recently among all the inputs to the post-synaptic node,
3. the outputs of the pre-synaptic node that caused the changed in the post-synaptic node's output.

These information are held by some variables named p_{ji}, q_{ji} and r_{ji} respectively. However, even a sequential EXOR problem could not be learned.

Then, for easy analysis, the algorithm is converted into the discrete time domain and only variable r_{ji} was used in the previous work[1]. Among the three variables, r_{ji} is particularly important because r_{ji} does not change when the output of post synaptic node does not change and is useful for the problems that need to memorize some past information before a long time lag. r_{ji} in the discrete time domain is updated at each time step as

$$r_{ji,t} = r_{ji,t-1}(1 - |\Delta x_{j,t}|) + f'(S_{j,t})x_{i,t}|\Delta x_{j,t}| \tag{1}$$

where $f'(S_j)$ is the derivate of the sigmoid function of j-th post-synaptic node, x_i, x_j is the output of the pre- and post-synaptic node respectively and $\Delta x_{j,t} = x_{j,t} - x_{j,t-1}$.

The important feature of r_{ji} is that, it holds the information about the output of the pre-synaptic node that caused the change of the pre-synaptic node's output. Each synaptic weight is updated as

$$\triangle\, w_{ji} = \eta \delta_j r_{ji} \tag{2}$$

where η is a learning rate and δ_j is propagated error of the post synaptic nodes.

2.1 Improvement of the PRL Method

Prevention of monotonous and biased oscillation in hidden nodes. By observing and analyzing the result from the previous work[1], it is shown that there is some monotonous oscillation in the change of hidden nodes' output for PRL during the learning phase. Fig.1 shows the change of the hidden node's output during the learning for BPTT and PRL whose connection weight to the output node is the largest. Almost half of the nodes' outputs in the hidden layer for PRL oscillate monotonously in some biased range of value. The net value of this output lies around the saturation range of the sigmoid function. It is well known that when the net value lies around the saturation range, the learning does not progress and it largely affects the learning performance.

Then, in order to accelerate learning, the output of hidden nodes is adjusted by moving the value to the vicinity of 0 when the output of the node oscillates around the saturation range of the sigmoid function. At first, the temporal average of the output is calculated in each epoch by

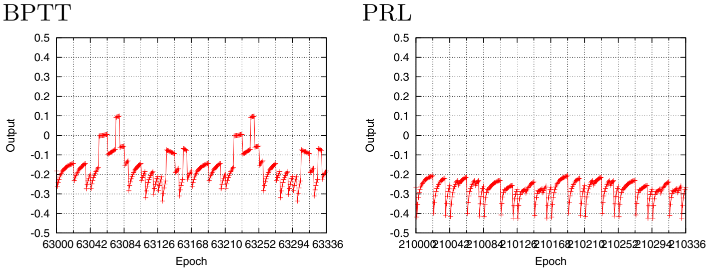


Fig. 1. The change of the output of the hidden node who has the largest connection weight to the output node for both methods. The left side is for BPTT, and the right one is for PRL.

$$\bar{o_j} = \frac{\sum_{\tau=0}^{T} o_{j,\tau}}{T}, \tag{3}$$

where $\overline{o_j}$ is the average of hidden jth output, T indicates the number of time steps for one epoch. Then the value of $\overline{o_{j,n}}$ is compared to average of $\overline{o_{j,n-1}}$ in the previous epoch according to the following equation.

$$\triangle\ o_{j,n} = \overline{o_{j,n}} - \overline{o_{j,n-1}}. \tag{4}$$

Then, if the difference of average value $|\triangle\ o_j|$ was below 0.1 and the state continued for 8 epochs, the hidden node's output is adjusted to the vicinity of 0 by the following equation before starting the next epoch. Here, we used the sigmoid function whose value range from -0.5 to 0.5.

$$o_{j,t} = o_{j,t} - \overline{o_{j,t-1}}. \tag{5}$$

3 Experimental Results

In this section, two different experiment results are presented to show the performance of the proposed extension PRL. The first experiment is the 3-bit parity problem as a benchmark test to show that modification of oscillated hidden neurons' output could improve the learning performance. The second experiment is a pattern classification task which is used to test whether PRL can perform in a more practical task.

Here, the network architecture used in this paper is an Elman-type RNN. Conventional BP method is used for the learning between hidden and output layers, and PRL is used for the learning between input and hidden layers.

3.1 3-Bit Parity Problem

In the preceding work[1], it was shown that 3-bit parity problem could be learned by PRL, but the learning performance is quite inferior to the BPTT method. RNN with 1 output, 20 hidden units and 4 input signals is used here. 3 of the inputs are the input signals to calculate the parity, and is given at t=5, 10, 15 sequentially. The other one is given to distinguish the starting time of one epoch and it is always 1 at t=0.

Table 1 summarizes some improving results of the PRL method. Here, successful learning is clarified when a squared error of less than 10^{-3} is continues for 100 patterns. In terms of success ratio, it is shown that PRL can perform better than before and similar to BPTT even though no trace back is done in this method. Although conventional BPTT performs better, in PRL, the average success iteration can be improved more than 50% compared to before.

In addition, Fig.2 shows the change of hidden node's output that have been adjusted to the vicinity of 0, resulting in faster learning than before.

Table 1. Comparison results of learning success and average success iterations

Learning rate of variable r	Learning success (/100times)	Average success iteration
Before modification		
1	99	33,107
2	85	29,737
4	42	33,666
10	7	17,331
After modification		
1	100	24,703
2	98	19,576
4	100	15,199
10	95	11,628
BPTT	100	6,297

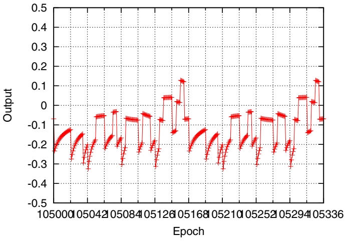


Fig. 2. The change of hidden node's output after introducing the method to move the value to the vicinity of 0

3.2 Pattern Classification Task

A critical test of the presented algorithm is to directly deal with high-dimensional, multimedia data, such as images or speech. Here, we carry out a handwritten digit recognition to investigate the performance of the proposed PRL.

The experiment was conducted on a digit database whose samples were collected by using a pen tablet as shown in Fig.3. Each of 10 different people wrote each of 10 numbers from 0 to 9 twice. Thus, 200 images were collected for training in total. In addition, we added 3 sample sets from 3 other people for test data as shown in Fig.3 to observe the generalization ability of both method. Each image has 100 rows and 100 columns, and each of 10,000 pixels has binary value. Considering the introduction of continuous-value inputs and effective generalization, the size of the image was reduced to 20 × 20 pixels by calculating the average of every 5 × 5 pixels. Then, in order to enter this digit image signals into a recurrent neural network, it is scanned column by column, resulting in 20 inputs per each step as shown in Fig.4. In addition, the number of steps is set to 20 in one iteration to represent 20 rows, and the training signal is provided only at t=20.

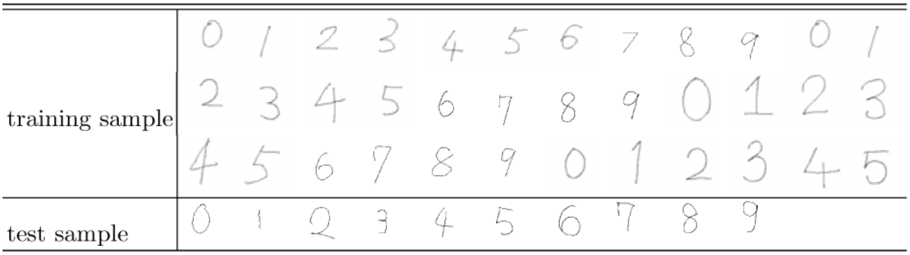


Fig. 3. Examples of handwritten 0∼9 digit numbers

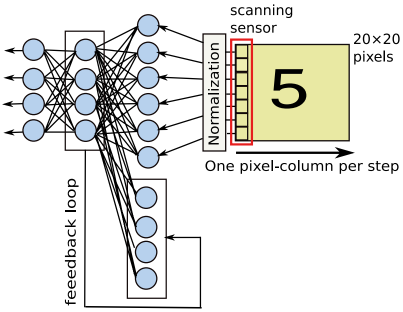


Fig. 4. A recurrent neural network with a handwritten digit image input by scanning column by column in each step

The task here is to classify the digit images into 10 classes. For instance, if an images of the number '1' is set as training data, the training signal for the 1*st* node in the output layer is 0.4, while the others will be -0.4. Furthermore, Table 2 shows the other parameter setup of the task.

Table 3 summarizes the comparison results for both methods. Here, the condition of successful learning is that the corresponding output to the presented image is the maximum among all the output nodes for every presented image. From the results, it is shown that the improved PRL could work even

Table 2. Parameter setups

Nodes in input layer	20 + hidden's layer nodes
Nodes in output layer	10
Nodes in hidden layer	40
Range of sigmoid function	-0.5∼0.5
Initial weight of (self-feedback)	4.0
Initial weight (non-self feed-back)	0.0
Initial weight (input to hidden layer)	random number of -1.0∼1.0
Initial weight (hidden to output layer)	0.0

Table 3. Comparison results of learning success ratio and average success iterations in the handwritten digit classification problem

Method	Learning success (/10 times)	Average success iteration
PRL	10	25,973
BPTT	10	14,666

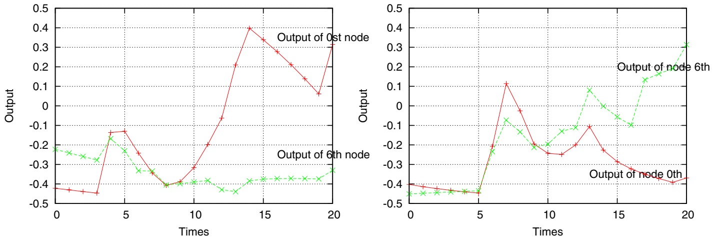


Fig. 5. The right side is the change in the outputs of $0th$ node and $6th$ node for an image of pattern '0' and the left side is the change for the image of pattern '6'

in the hand-writing recognition task with almost continuous-value inputs that is more difficult compared to the parity problem. It is also shown here that in terms of success ratio, PRL can perform almost as good as BPTT, but conventional BPTT still performs better in the number of average success iteration.

In order to show how the RNN classifies the images, two samples output changes in one epoch are shown in Fig.5 for presence of '0' and '6'. The left half of '0' and '6' is similar to each other. It can be seen that the change of output in $0th$ and $6th$ node is similar at the early times, but the corresponding output is increases after the latter half of the epoch.

In order to observe the generalization ability of both method, the performance for the test data is examined. It is shown that all of the test data could be classified by the PRL method while 15 from 30 images in BPTT. PRL could recognize all the images in test samples, though further experiments are required to validate the results of generalization ability between both methods.

4 Conclusion

One extension are introduced here in order to improve performance of the PRL method that capable to keep the order of memory size and computational cost as low as $O(n^2)$, which are not realized by BPTT and RTRL. By preventing the oscillated hidden node's output from being biased to the saturation of the sigmoid function, it was shown that the learning performance of PRL in 3-bit parity problem could be improved. Furthermore, it was shown that PRL works in scanned digit pattern classification task that is more practical than the parity problem. However, since PRL is still inferior to BPTT in the average number of iteration for learning, future investigations for further improvement and application to more practical tasks are required.

Acknowledgement. A part of this research was supported by JSPS Grant in-Aid for Scientific Research and ♯19300070.

References

1. Samsudin, M.F., Hirose, T., Shibata, K.: Practical Recurrent Learning (PRL) in the Discrete Time Domain. In: Ishikawa, M., Doya, K., Miyamoto, H., Yamakawa, T. (eds.) ICONIP 2007, Part I. LNCS, vol. 4984, pp. 228–237. Springer, Heidelberg (2008)
2. Shibata, K., Okabe, Y., Ito, K.: Simple Learning Algorithm for Recurrent Networks to Realize Short-Term Memories. In: Proc. of IJCNN Intl. Joint Conf. on Neural Network, vol. 98, pp. 2367–2372 (1998)
3. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by errorpropagating. In: Parallel Distributed Processing, vol. 1, pp. 318–362. MIT Press, Cambridge (1986)
4. Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural network. Neural Computation 1, 270–280 (1989)
5. Song, Q., Wu, Y., Soh, Y.C.: Robust Adaptive Gradient-Descent Training Algorithm for Recurrent Neural Networks in Discrete Time Domain. IEEE Transactions on Neural Networks 19(11), 1841–1853 (2008)
6. Hochreiter, S., Schimidhuber, J.: Long Short Term Memory. Neural Computation 9, 1735–1780 (1997)

An Automatic Intelligent Language Classifier

Brijesh Verma[1], Hong Lee[1], and John Zakos[2]

[1] School of Computing Sciences, CQUniversity
Rockhampton, Queensland, Australia
{B.Verma,H.Lee1}@cqu.edu.au
[2] MyCyberTwin, Gold Coast, Queensland, Australia

Abstract. The paper presents a novel sentence-based language classifier that accepts a sentence as input and produces a confidence value for each target language. The proposed classifier incorporates Unicode based features and a neural network. The three features Unicode, exclusive Unicode and word matching score are extracted and fed to a neural network for obtaining a final confidence value. The word matching score is calculated by matching words in an input sentence against a common word list for each target language. In a common word list, the most frequently used words for each language are statistically collected and a database is created. The preliminary experiments were performed using test samples from web documents for languages such as English, German, Polish, French, Spanish, Chinese, Japanese and Korean. The classification accuracy of 98.88% has been achieved on a small database.

1 Introduction

Automatic language classification systems are needed in many real world applications such as web based communication, multilingual document classification, medical cross-language text retrieval systems, helpdesk call routing and spoken language classification just to mention a few.

Automatic language classification is the problem of identifying in which language a given sample text has been written. Living in a global community, we are surrounded by multi-lingual environments such as web documents, speeches, etc. Especially, global advances in the Internet communities have imposed a great deal of importance for language classification problem due to the huge amount of web documents published in multi-languages. Successful research outcomes can affect many industrial sectors. A multi language translation technique [1, 2] is one of the examples, where the input language needs to be classified prior to the translation to a target language. Also, the language identification plays a key role in the internet search engines by identifying the language of the search keys [3]. Researchers have found [4] that text-to-speech applications heavily depend on the language identification performances in multi-lingual environments.

Language classification tasks based on the written mono text (single language document) has been regarded as a relatively simple problem for small number of languages and when a large amount of sample texts in the identification stage are available. However, the task of language classification is very difficult and challenging when we have multi-language documents and large number of languages to

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 639–646, 2009.

classify. The complexity of the problem solving significantly increases [5] with the size of input text.

The main goal of the research presented in this paper is to investigate a novel classifier that accepts a sentence including multilingual/small sentence as input and provide a confidence value for each language. The paper is divided into five sections. Section 2 presents existing techniques for language classifiers, limitations and difficulties. Section 3 presents the proposed research methodology. The experimental results and analysis are presented in Section 4. Finally, a conclusion is presented in Section 5.

2 Background

The standard framework involved in language identification is modeling and classification. In language modeling stage, the most discriminative features of each target language is extracted and stored in its language model. During classification, similar feature extraction process is performed on input texts. Based on the models of each language and input text, the distance of similarity or dissimilarity is measured and the input text is identified according to the score. In [6], a language identification system has been presented which can achieve accuracy of 93% with as little as a three-word input.

There has been some research conducted in the area of automatic classification of languages and some papers have been published in recent years. In [3], an approach is proposed which can classify input texts' language by finding the maximum frequency of input words in each dictionary of Spanish, French, English, Portuguese, German and Italian. To identify the input language, heuristics are employed into the decision making process. The methodology is effective to classify input texts' language as accurate as 88% on randomly selected web pages and 99% on randomly selected well-formatted texts. In [5], a decision tree scheme for common letters of language in documents is used to identify Arabic from Persian. The decision tree is defined as a series of questions about the context of the current letter. If a common but discriminant letter from the other language is found, the classification is made on the incident. The experiment result shows that average of around 98.8% accuracy was achieved to identify 240 web documents (120 for Arabic and 120 for Persian). In [6], each language is modeled from a corpus of training documents on features extracted based on common words and N-gram methods. The features extracted by the common words are the probability distribution of the frequency of the most common words in the training documents in a language. Likewise, features of character N-gram is measured to reflect the frequency score and the rank of N-gram instances are stored. During classification, rather than modeling the whole input text, features of random subsections of the input texts are extracted to minimize the computational time. The random sampling is performed until the standard error of the random samples is larger than a threshold. The Monte Carlo method with N-gram and common words was tested on Danish, Dutch, English, French, German, Italian, Norwegian, Portuguese, Spanish, and Swedish from ECI database. However, it doesn't report the numerical data on the performance of the classification apart from the comparative graph between difference methods. In [8], two identification methods, enhanced N-gram probabilities and decision tree are proposed to compare the performance of classification accuracy. The authors enhanced N-gram feature extraction technique by decomposing each word into three parts, head, body and tail. The decision trees are to identify the most likely language for each letter in the input word. The experiment results on local in-house guest names in four languages reported that 71.8% and 66.1% average identification accuracies were achieved by N-gram and decision tree methods accordingly. Vector-space based identification approach was proposed in [9] for 13 Latin character based languages. Features included in vectors were N-gram frequencies and word sizes with inverse document frequency weight incorporated. Between models and an input, cosine values are calculated and used to classify the input text. Experiment results report various performance accuracies depending on the input text size, which produced 100% accuracy on web documents with 1000 bytes. In [10], the authors incorporated feature extraction technique of the common words in a language, known as stop words like 'the', 'of' and 'to', to identify the language from scanned document images written in multi-lingual environments. In their research, the stop words, their frequency and word shape code are used as key feature vectors to classify the language which input documents were written in. The approach was as effective as 96.75% of accuracy rate at best on locally prepared database. In [11], Artemenko et al. evaluated performances of four different identification methodologies in two separate experiments of mono-lingual and multi-lingual web documents on 8 languages. Identification methods used in the experiments were Vector space cosine similarity, 'out of place' similarity between rankings and Bayesian classifier on N-gram feature spaces. A word frequency based classification was added to the comparison. The research inferred that N-gram based approach outperforms the word frequency based methods for short texts. The researchers were able to achieve 100% and 97% accuracies on mono and multi lingual documents accordingly. In [12], an approach was proposed which can count common words and character sequences of N-gram methods for a language. Then, the frequency was used as the key information to distinguish the input documents against models. The performance was measured on Europarl corpus test sets, and was satisfactory, 97.9% on German language was achieved. In [13], an algorithm is presented which extends the common N-gram corpus analysis complemented with heuristics. Classification was to measure the similarity between input text and model languages. The literature reports the performance on 12 languages of 6000 web documents was 100% accurate at most. Rendering character sequence into HMM language model was manipulated as a key ingredient for language classification task in [14]. The identification accuracy of 95% was achieved in their proposal. In [15], term frequency and its weight by entropy method over documents were used as feature for neural networks to categorize the web documents. In [16], an approach was proposed which uses tri-gram and frequency language modelling technique to identify the origin of names written-in-Latin, Japanese, Chinese and English. An accuracy of 92% was achieved to distinguish Japanese names from the others.

3 Proposed Research Methodology

The proposed research methodology is described in details in this section. Foremost, an overview of the proposed technique is presented, followed by analysis of language specific Unicode. The proposed feature extraction and classification algorithms are described at the end of this section.

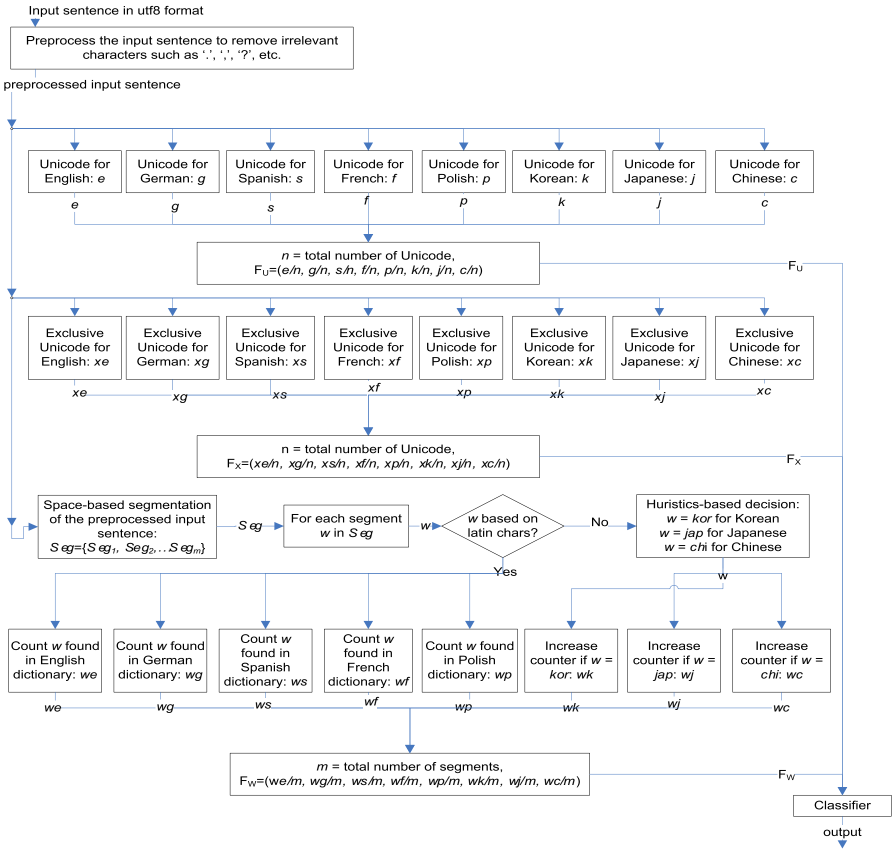

Input sentence in utf8 format
Preprocess the input sentence to remove irrelevant characters such as '.', ',', '?', etc.
preprocessed input sentence
Unicode for English: e
Unicode for German: g
Unicode for Spanish: s
Unicode for French: f
Unicode for Polish: p
Unicode for Korean: k
Unicode for Japanese: j
Unicode for Chinese: c
e
g
s
f
p
k
j
c
n = total number of Unicode,
F_U=(e/n, g/n, s/n, f/n, p/n, k/n, j/n, c/n)
F_U
Exclusive Unicode for English: xe
Exclusive Unicode for German: xg
Exclusive Unicode for Spanish: xs
Exclusive Unicode for French: xf
Exclusive Unicode for Polish: xp
Exclusive Unicode for Korean: xk
Exclusive Unicode for Japanese: xj
Exclusive Unicode for Chinese: xc
xe
xg
xs
xf
xp
xk
xj
xc
n = total number of Unicode,
F_X=(xe/n, xg/n, xs/n, xf/n, xp/n, xk/n, xj/n, xc/n)
F_X
Space-based segmentation of the preprocessed input sentence: Seg={Seg_1, Seg_2,...Seg_m}
Seg
For each segment w in Seg
w
w based on latin chars?
No
Yes
Huristics-based decision: w = kor for Korean w = jap for Japanese w = chi for Chinese
w
Count w found in English dictionary: we
Count w found in German dictionary: wg
Count w found in Spanish dictionary: ws
Count w found in French dictionary: wf
Count w found in Polish dictionary: wp
Increase counter if w = kor: wk
Increase counter if w = jap: wj
Increase counter if w = chi: wc
we
wg
ws
wf
wp
wk
wj
wc
m = total number of segments,
F_W=(we/m, wg/m, ws/m, wf/m, wp/m, wk/m, wj/m, wc/m)
F_W
Classifier
output

Fig. 1. Proposed methodology

3.1 Overview

The proposed approach shown in Fig. 1 takes an UTF8 formatted sentence as an input. The irrelevant Unicodes from the input sentence, are removed through a preprocessing module before feature extraction. Based on the preprocessed input, Unicodes for each language are counted and divided by the total number of Unicodes. The second feature is to extract and count the language specific Unicodes for each language. Again, the count for exclusive Unicode of each language is divided by the total number of Unicodes in the preprocessed input sentence. The final feature is related to identifying each segment or word separated by a space. There are two different methodologies suggested for identification of each segment or word. Firstly, Latin character-based languages like English, German, Polish, French, Spanish, etc. put spaces between words. However, it is not true for Chinese, Japanese, Korean, etc. to distinguish words by spaces. So, it is more appropriate to call a component in a sentence separated by spaces a "segment". To identify each segment composed of Latin characters, a dictionary matching method is proposed. Heuristics are employed to identify each segment composed of Unicodes from Chinese, Japanese and Korean. Finally, the features are fed to a classifier to decide which language the input sentence belongs to and provide confidence values.

3.2 Unicode and UTF8

Unicode is a standard representation of characters to be expressed in computing [17]. UTF8 (8-bit Unicode Transformation Format) is a preferred protocol to encode the Unicode characters for storing or streaming them electronically [18, 19]. Characters for all languages are defined as ranges in UTF8 format [20].

3.3 Feature Extraction

The proposed methodology utilizes three features related to Unicode components in the pre-processed sentence. The three features are Unicode, exclusive Unicode and segments for each language.

3.3.1 Unicode Feature

$F_U = \{u_1, u_2, \ldots, u_n\}$, $u_m = \frac{g(m)}{t}$, m = 1…n, where *n* is the number of languages, *g(m)* is the total number of Unicodes for language *m*, and *t* is the total number of Unicode in an input sentence.

Example:

Input sentence: ‘My name is 이상민 in Korean’

Unicode distribution:

English, German, French, Spanish, and Polish: M, y, n, a, m, e, i, s, i, n, k, o, r, e, a, n (Total: 16)

Chinese and Japanese: 0

Korean: 이, 상, 민 (Total: 3)

Total Unicode in the input sentence: 19

$F_U = \{16/19, 16/19, 16/19, 16/19, 16/19, 0, 0, 3/19\}$

3.3.2 Exclusive Unicode Feature

$F_X = \{u_1, u_2, \ldots, u_n\}$, $u_m = \frac{g(m)}{t}$, m = 1…n, where *n* is the number of languages, *g(m)* is the total number of exclusive Unicode for language *m*, and *t* is the total number of Unicode in an input sentence.

3.3.3 Segment feature

$F_W = \{S_1, S_2, \ldots, S_n\}$, $S_m = \frac{f(m)}{y}$, m = 1…n, where *n* is the number of languages, *y* is the total number of space-separated segments/words, and *f(m)* is the total number of space-separated segments/words identified by dictionary matching in a pre-processed input sentence.

3.4 Classification

The weighted sum feature classification and neural network based classification as described below have been investigated in this research.

3.4.1 Weighted Sum Feature Classification (WSFC)

The three features described in previous sections are extracted from the input sentence. The features are multiplied by a preset weight and sum of weighted features is calculated. The language with highest weighted feature value is selected.

3.4.2 Neural Classification (NC)

An overview of neural classification process is shown below in Fig. 2. The three features are extracted from the input sentence and fed to a neural classifier. The classifier fuses the features and gives the final confidence for each language. The final output contains the total score for each language. The neural network based classifier is trained using artificially generated training set before it is used for testing.

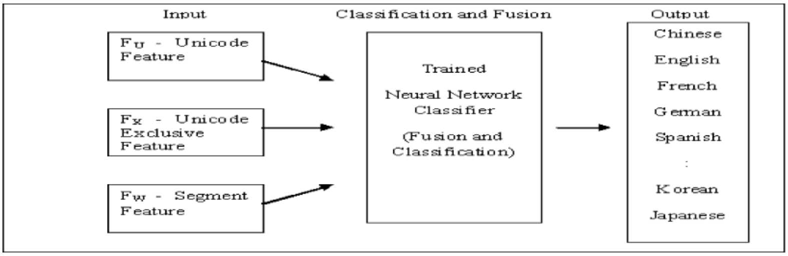


Fig. 2. Overview of neural network classification process

4 Experiments and Results

The proposed methodology has been implemented in Java programming language. The experiments were conducted using weighted sum feature classification (WSFC) and neural classification (NC).

A small database of sentences with less than 10 words taken from web pages has been created. A news article for each language is selected and input samples for testing has been prepared by segmenting the article into sentences by finding a period symbol "." at the end of sentence. One hundred sentences for each language were collected, which give the total of eighty hundred sentences, and stored in an input file with UTF8 format.

The classification accuracies for experiments are shown in Table 1. The proposed approach has been compared to other methods in the literature. The proposed approach shows the similar performance over methodologies in [5, 8, 11, 12], but they are lower than the results from [9, 13]. However, considering the input language mode and the size of the input data, it is fair to conclude that the proposed approach were competitive to the existing approaches. Unlike the proposed approach, experimental results from [9] used longer input data size. The method in [11] has achieved the higher accuracy than the proposed approach on only mono input lingual mode. Finally, the approach in [13] achieved 100% accuracy on only English input documents.

Table 1. Experimental results

Technique	Number of Sentences	Accuracy on Test Data [%]
Proposed approach with WSFC	800	98.88
Proposed approach with NC		98.88

5 Conclusions and Future Research

In this paper, a novel approach for language classification has been presented and investigated. UTF8 encoding scheme has been used to construct the features for classification. The Unicode, exclusive Unicode and word matching score features in conjunction with a neural network are used to classify a language of an input sentence. Word matching score was extracted against a common word list of each language, rather than full length dictionaries, to simplify the computational searching cost. The experiments with the proposed approach produced very competitive results, considering the limited length of input sentences. In our future research, the focus will be on improving the training data for neural network and testing on shorter sentences.

Acknowledgements

This research work is supported by CQUniversity's industry collaborative small research grant and MyCyberTwin.

References

1. Artemenko, O., Mandl, T., Shramko, M., Womser-Hacker, C.: Evaluation of a language identification system for mono- and multilingual text documents. In: Proceedings of the 2006 ACM symposium on applied computing, pp. 859–860 (2006)
2. Dunning, T.: Statistical identification of language, Technical report CRL MCCS-94-273, New Mexico State University, Computing Research Lab (March 1994)
3. Hakkinen, J., Tian, J.: N-gram and decision tree based language identification for written words. In: IEEE workshop on automatic speech recognition and understanding (ASRU 2001), pp. 335–338 (2001)
4. Lins, R., Goncalves, P.: Automatic language identification of written texts. In: Proceedings of the 2004 ACM symposium on applied computing, pp. 1128–1133 (2004)
5. Lu, S., Tan, C.L.: Retrieval of machine-printed Latin documents through word shape coding. Pattern Recognition 41(5), 1799–1809 (2008)
6. Martins, B., Silva, M.: Language identification in web pages. In: Proceedings of the 2005 ACM symposium on applied computing, pp. 764–768 (2005)
7. McNamee, P.: Language identification: a solved problem suitable for undergraduate instruction. J. Comput. Small Coll. 20(3), 94–101 (2005)
8. Muthusamy, Y., Barnard, E., Cole, R.: Automatic language identification: a review/tutorial. IEEE Signal Processing 11, 33–41 (1994)

9. Poutsma, A.: Applying monte carlo techniques to language identification. In: Proceedings of computational linguistics in the Netherlands (CLIN), vol. 45, pp. 179–189 (2001)
10. Prager, J.L.: Language identification for multilingual documents. In: Proceedings of the 32nd annual Hawaii international conference on system sciences, vol. 2, p. 2035 (1999)
11. Qu, Y., Grefenstette, G.: Finding ideographic representations of Japanese names written in Latin script via language identification and corpus validation. In: Proceedings of the 42nd annual meeting on association for computational linguistics, p. 183 (2004)
12. Romsdorfer, H., Pfister, B.: Text analysis and language identification for polyglot text-to-speech synthesis. Speech communication 49(9), 697–724 (2007)
13. Selamat, A., Omatu, S.: Web page feature selection and classification using neural networks. Information sciences 158, 69–88 (2004)
14. Selamat, A., Ching, N.C., Mikami, Y.: Arabic script web documents language identification using decision tree-ARTMAP model. In: IEEE International conference on convergence information technology, pp. 721–726 (2007)
15. The Unicode Consortium. The Unicode Standard, Version 5.0, 5th edn. Addison-Wesley Professional, Reading (2006)
16. Unicode (June 26, 2008), Wikipedia, http://en.wikipedia.org/wiki/Unicode (Retrieved June 27, 2008)
17. Unicode/UTF-8-character table, UTF8-CharTable, http://www.utf8-chartable.de/unicode-utf8-table.pl (Retrieved June 26, 2008)
18. UTF-8 (June 24, 2008). Wikipedia: http://en.wikipedia.org/wiki/UTF-8 (Retrieved June 26, 2008)
19. Xafopoulos, A., Kotropoulos, C., Almpanidis, G., Pitas, I.: Language identification in web documents using discrete HMMs. Pattern recognition 37(3), 583–594 (2004)
20. Yergeau, F.: UTF-8, a transformation format of ISO 10646. In: RFC 3629, internet engineering task force (2003)

Gender Classification by Combining Facial and Hair Information

Xiao-Chen Lian and Bao-Liang Lu*

Department of Computer Science and Engineering
MOE-Microsoft Key Lab. for Intelligent Computing and Intelligent Systems
Shanghai Jiao Tong University, Shanghai 200240, China
bllu@sjtu.edu.cn

Abstract. Most of the existing gender classification approaches are based on face appearance only. In this paper, we present a gender classification system that integrates face and hair features. Instead of using the whole face we extract features from eyes, nose and mouth regions with Maximum Margin Criterion (MMC), and the hair feature is represented by a fragment-based encoding. We use Support Vector Machines with probabilistic output (SVM-PO) as individual classifiers. Fuzzy integration based classifier combination mechanism is used to fusing the four different classifiers on eyes, nose, mouth and hair respectively. The experimental results show that the MMC outperforms Principal Component Analysis and Fisher Discriminant Analysis and incorporating hair feature improves gender classification performance.

1 Introduction

Gender classification is a high-level field in computer vision and is very important since its widespread applications in human-computer interaction and services that depend on it, such as demographics. There are many gender classification methods based on appearance [1,2,3,4,5]. Most of the existing approaches only utilize the internal facial information. We've conducted a psychological experiment: 16 participants were asked to do the gender classification task on 200 images. They were divided into two groups: Group A were fed with complete faces, and Group B were given images with only inner faces (See Fig. 1(a)). The accuracy of Group A is 100%, while the average accuracy of Group B is 92.5%. Fig. 1(b) shows images misclassified by more than 5 people of Group B. From this experiment we conclude that hair, can provide discriminative clues, especially when the face is somewhat neutral.

Following the above discussion, it is natural to expect better performance by combining face and hair information. In [6], Ueki *et al.* divided an image into face, hair, and clothing regions, and a model was learned independently for each region. The final classification of the face image was made by combining these models using a Bayesian approach.

* Corresponding author. This research was partially supported by the National Natural Science Foundation of China via the grant NSFC 60773090.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 647–654, 2009.

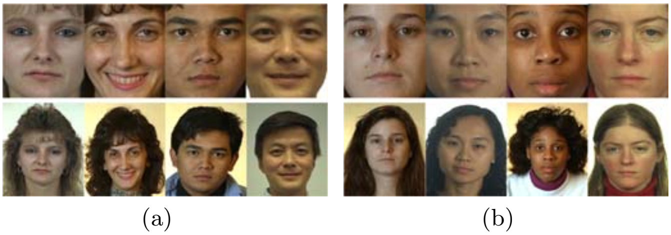


Fig. 1. Psychological experiment about gender classification: (a)experimental data; (b)images misclassified by more than 5 people of Group B

In this paper, we propose a gender classification system which makes decisions by integrating face and hair information. To represent the face information, we only use the local facial features, that is, we extract the eyes, nose and mouth regions of a face and discard the rest part. Hair is hard to represent due to its large variation. Lapedriza *et al.* [7] proposed a fragment-based method which is relatively insensitive to illumination changes. We adopt this method and modify it to reduce the influence of background.

In the literature of pattern recognition, there are generally two categories of information integration approaches. One is feature combination and the other is classifier combination. We choose a fuzzy-integration-based classifier combination method. Four classifiers are trained for eyes, nose, mouth and hair features separately. Fuzzy integral is then used to combine the decisions of these classifiers into a single composite score with respect to a designated fuzzy measure. The whole process is illustrated in Fig. 2.

The remaining part of the paper is organized as follows: in section 2, face representation based on local features is introduced. In section 3, we describe the hair feature extraction method in details. Experiments and analysis are conducted in section 4, followed by conclusion and discussion in the last section.

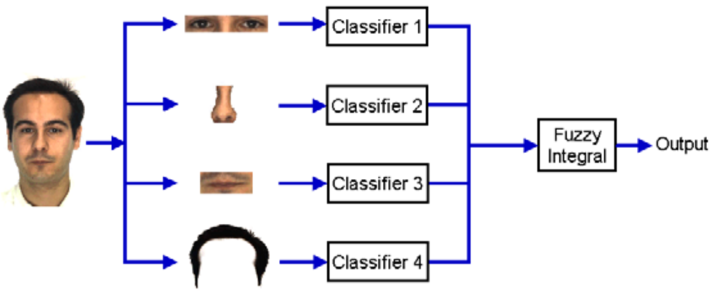


Fig. 2. The proposed gender classification system by combining four different SVMs

2 Facial Feature Representation

In this section, we introduce the local-feature-based face representation method. Unlike most of the methods which use the whole face, we only consider the facial components: eyes, nose and mouth. Local features are believed very robust to the variations of facial expression, illumination, and occlusion [8]. Furthermore, psychological experiments showed that individual features (brows, eyes, nose, mouth, and chin), when seen in isolation, carried some information about gender [9].

To obtain the facial components, we adopt Active Shape Model (ASM) [10], a statistical model of the shape of the deformable object, to get the locations of eyes, nose and mouth, and then extract the rectangles centered at them respectively.

Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are two most widely used linear subspace learning approaches in face recognition. PCA simply performs a coordinate rotation that aligns the transformed axes with the directions of maximum variance. It ignores the class label information and therefore is not optimal for general classification tasks. LDA, or Fisher Discriminant Analysis (FDA), is to pursue a low dimensional subspace that can best discriminate samples from different classes It requires that the within-class scatter matrix is not singular. Hence LDA is not stable facing the small sample size problem which is very common in face recognition and gender classification tasks. When applying LDA, we often have to preprocess the data with PCA.

Maximum Margin Criterion (MMC) is a recently proposed supervised feature extraction criterion [11]. Using the same notation as LDA, the goal of MMC is to maximize the criterion $J(W) = W^T(S_b - S_w)W$. The projection matrix W can be obtained by solving the following eigenvector decomposition problem:

$$(S_b - S_w)w = \lambda w \tag{1}$$

where w is the desired projection and S_b,S_w are between- and within- class scatter matrix respectively. Note that MMC does not have inverse operation and hence does not suffer from the small sample size problem.

We apply MMC to the extracted facial components. As gender classification is a 2-class problem, the resulted vector is one dimension.

3 Hair Feature Representation

Hair is represented by a fragment-based encoding. We describe the algorithm briefly. Given a collection of aligned face images C and a collection of non-face images $\overline{C}$, a representative set of image fragments called Building Blocks set (BBS) is constructed by select K most discriminative fragments among all fragments generated from regions in each image of C that contains hair. A fragments is discriminative if it appears often in C but seldom in $\overline{C}$. Some examples are shown in Fig. 3. To represent an unseen image I from BBS, for each element f_i of

BBS a black image B_i with the same size as I is constructed. Then f_i is located on B_i where it best matches. The similarity of two images p and f is computed by Normalized Cross Correlation (NCC). These B_i images constitute the basis and the image I is represented by linear combination of them $I \simeq \sum_{i=1}^{K} w_i B_i$. To make the combination having sense, the coefficients w_i are required non-negative. It is apparently a quadric programming aiming to minimizing the reconstruction error $(I - \sum_{i=1}^{K} w_i B_i)^2$ under the constrains $w_i \geq 0 (i = 1, \ldots, K)$. The resulted coefficient vector $W = \{w_1, \ldots, w_K\}$ encodes the hair information.

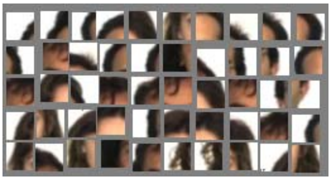


Fig. 3. Some of the building blocks

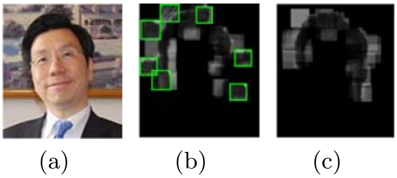


Fig. 4. Illustration of the improvement

This algorithm has a problem: some fragments might be put on the background region: as shown in Fig. 4(b), there are many misplaced fragments (ones marked with green rectangles) produced by the original method with misplaced. This happens when a fragment is more similar to background than to hair in that image. Based on our experience with the implementation, the mean value of all fragments' NCC values with non-face images of $\overline{C}$ is 0.563 and the standard deviation is 0.001, and the mean value of all fragments' NCC values with face images of C is 0.818 and the standard deviation is 0.003. That means the distribution of NCC values with background and the distribution of those with hair region are separated. Therefore we discard the B_is with NCC values less than 0.750. Our modification reduces the influence of complex background (Fig. 4(c)).

4 Classifier Combination Mechanism

The concept of fuzzy integral was originally introduced by Sugeno [12] and has become increasingly popular for multi-attribute classification [13]. In this section we briefly review the concepts of fuzzy measure and fuzzy integral, and then describe how it can be applied to classifier combination.

4.1 Fuzzy Measure and Fuzzy Integral

Fuzzy measure is a extension of the classical measure where additivity property is relaxed.

Definition 1. *A fuzzy measure μ defined on $X = \{x_1, \dots, x_n\}$ is a set function $\mu : \mathcal{P}(X) \rightarrow [0,1]$ satisfying*
(1) $\mu(\emptyset) = 0$, $\mu(X) = 1$.
(2) $A \subseteq B \Rightarrow \mu(A) \leq \mu(B)$
$\mathcal{P}(C)$ indicates the power set of X.

Fuzzy integrals are integrals of a real function with respect to a fuzzy measure (by analogy with Lebesgue integral). There are several forms of fuzzy integrals. The one adopted here is the Choquet integral proposed by Murofushi and Sugeno [14].

Definition 2. *Let μ be a fuzzy measure on X. The discrete Choquet integral of a function $f : X \rightarrow \mathbb{R}^+$ with respect to μ is defined by*

$$C_\mu(f(x_1), \dots, f(x_n)) \triangleq \sum_{i=1}^{n} \left(f(x_{(i)}) - f(x_{(i-1)})\right) \mu(S_{(i)}) \qquad (2)$$

where $\cdot_{(i)}$ indicates that the indices have been permuted so that $0 \leq f(x_{(1)}) \leq \cdots \leq f(x_{(n)}) \leq 1$. And $S_{(i)} \triangleq \{x_{(i)} \dots, x_{(n)}\}$.

4.2 Classification by Fuzzy Integral

Fuzzy integral has two advantages. One is that a number of combination mechanism are special cases of it, e.g., weighted sum, min and max rules; The other is that we can represent the importance of individual classifier and interactions (redundancy and synergy) among any subset of the classifiers using an appropriate fuzzy measure.

We describe how fuzzy integral is applied in classification tasks. Suppose $T = \{t_1, \dots, t_m\}$ is a set of given classes. Let X be the set of classifiers and μ, a fuzzy measure defined on X, represent the contribution of each subset of X in final decision. For an unknown sample A, let $h_j(x_i)$ be the confidence of "A belongs to class t_j" given by classifier x_i. Then the global confidence of "A belongs to t_j" is given by $C_\mu(h_j(x_1), \dots, h_j(x_n))$. We denote it by $C_\mu^j(A)$. Finally, A is given the class with highest confidence.

As mentioned above, fuzzy measure represents the contributions of classifier subsets, hence determining the fuzzy measure is a crucial step when applying fuzzy integral. For a n-classifier combination problem, we have to determine $2^n - 2$ values (fuzzy measure on empty set and X are known). These values can be learned from training data. Suppose that $(z_k, y_k), k = 1, \dots, l$ are learning data of a 2-class problem. For simplicity, we assume that $y_k = 1$ for $k = 1, \dots, l_1$ and $y_k = -1$ for $k = l_1 + 1, \dots, l_2$ and $l = l_1 + l_2$. We try to identify the best fuzzy measure μ so that the squared error is minimized

$$J = \sum_{i=1}^{l_1} \left(C_\mu^1(z_i) - c_\mu^2(z_i) - 1\right)^2 + \sum_{i=l}^{l_2} \left(C_\mu^2(z_{l+i}) - c_\mu^1(z_{l+i}) - 1\right)^2 \qquad (3)$$

It can be solved under quadratic program form.

Fuzzy integral requires that the output of the classifiers should be some confidence form. We choose SVM-PO with RBF kernel[15] (for the one-dimension facial features, it is equivalent to the posterior probability estimation under the assumption that the data distribution for each gender is Gaussian).

5 Experiments

The experimental data come from frontal faces of four database with different races (See Tab. 1). We first evaluated PCA, FDA and MMC using face features (See Tab. 2). The signal-to-noise ratio of PCA was set to be 9 : 1. And the data were preprocessed by PCA for FDA. MMC utilizes the label information and therefore outperformed the other two.

Our hair feature extraction method are compared the other two methods. The first one [18] extracted the geometric features from profile of hair and the other one [6] applied gaussian mixture model on the 32 × 32 hair-only images. The result is shown on Tab. 3.

To investigate the relationship between fuzzy measure values and classifiers' performance, we randomly picked one fifth of training data as the validation set and computed the fuzzy measure by minimizing (3) on the validation set. Accuracy of individual classifiers on validation set is shown in Tab. 4 (Here $C_{(\cdot)}$ denotes the classifier using a particular feature). C_{eyes} performs best and the accuracy of C_{nose} is lowest. Intuitively, in the final decision the contributions of these classifiers should be proportional to their accuracy. Fuzzy measures illustrated in Tab. 5 support this intuition: the order of fuzzy measure values on

Table 1. Experimental data

Training data			Test data		
Database	female	male	Database	female	male
FERET	490	718	Postech PF01 [17]	51	54
CAS-PEAL [16]	445	595	AR	45	64
Total	935	1313	Total	96	118

Table 2. Gender classification performance of PCA, FDL and MMC

	face	eyes	nose	mouth
SVM	87.85%	85.98%	66.82%	82.71%
PCA	84.03%	84.04%	64.79%	80.75%
FDA	81.69%	83.10%	61.03%	79.34%
MMC	87.32%	85.86%	65.38%	81.69%

Table 3. Gender classification performance using hair feature extraction methods

Methods	Ours	ICCV'05	ICPR'04
Accuracy	81.60%	79.59%	72.66%

Table 4. Accuracy of individual classifiers on validation set

Classifier	C_{eyes}	C_{nose}	C_{mouth}	C_{hair}
Accuarcy	84.25%	70.94%	78.94%	75.41%

Table 5. Fuzzy measure values of classifier subsets

Classifier subset	μ	Classifier subset	μ
$\{\emptyset\}$	0.000	$\{C_{\text{eyes}}\}$	0.691
$\{C_{\text{nose}}\}$	0.100	$\{C_{\text{mouth}}\}$	0.640
$\{C_{\text{hair}}\}$	0.148	$\{C_{\text{eyes}},C_{\text{nose}}\}$	0.691
$\{C_{\text{eyes}},C_{\text{mouth}}\}$	0.909	$\{C_{\text{eyes}}, C_{\text{hair}}\}$	0.883
$\{C_{\text{nose}},C_{\text{mouth}}\}$	0.640	$\{C_{\text{nose}},C_{\text{hair}}\}$	0.148
$\{C_{\text{mouth}},C_{\text{hair}}\}$	0.742	$\{C_{\text{eyes}},C_{\text{nose}},C_{\text{mouth}}\}$	0.909
$\{C_{\text{eyes}},C_{\text{nose}},C_{\text{hair}}\}$	0.883	$\{C_{\text{eyes}},C_{\text{mouth}},C_{\text{hair}}\}$	1.000
$\{C_{\text{nose}},C_{\text{mouth}}C_{\text{hair}}\}$	0.742	$\{C_{\text{eyes}},C_{\text{nose}},C_{\text{mouth}},C_{\text{hair}}\}$	1.000

Table 6. Accuracy of gender classification with various methods

Fuzzy integral	Weighted sum	Product	CCA	Face
90.61%	88.73%	87.63%	74.76%	87.32%

individual classifiers is $\mu(\{C_{\text{eyes}}\}) > \mu(\{C_{\text{mouth}}\}) > \mu(\{C_{\text{hair}}\}) > \mu(\{C_{\text{nose}}\})$; The better a individual classifier performs, the larger fuzzy measure value on a subset containing it we obtain.

Finally, we compared gender classification performance of our method with the methods using only face information. We also compared different combination methods. Besides fuzzy integral, we chose two classifier combination mechanisms–weighted sum and product rule –and one widely used feature combination method exploiting Canonical Correlation Analysis (CCA). The results are shown in Tab. 6. The last column is the performance with MMC using only face features. Fuzzy integral produced highest accuracy among these combination methods. And integrating face and hair feature through fuzzy integral improved the gender classification performance.

6 Conclusions and Future Work

One major contribution of this paper is integrating hair and face features through fuzzy-integration-based classifier combination approach. Moreover, instead of using the whole face, we extract features of facial components by MMC. The experimental results show that MMC outperforms PCA and FDA, and the gender classification benefits from incorporation of hair. A future extension of our work is to utilize clothing information. We plan to extract the profile, texture and color distribution information.

References

1. Golomb, B., Lawrence, D., Sejnowski, T.: Sexnet: A Neural Network Identifies Sex from Human Faces. Advances in Neural Information Processing Systems 3, 572–577 (1991)
2. Xia, B., Sun, H., Lu, B.L.: Multi-view Gender Classification based on Local Gabor Binary Mapping Pattern and Support Vector Machiness. In: IJCNN 2008, HongKong, China, pp. 3388–3395 (2008)
3. Lian, H.C., Lu, B.L.: Multi-View Gender Classification Using Local Binary Patterns and Support Vector Machines. International Journal of Neural System 17(6), 479–487 (2007)
4. Moghaddam, B., Yang, M.: Learning Gender with Support Faces. IEEE Transactions on Pattern Analysis and Machine Intelligence, 707–711 (2002)
5. Kim, H., Kim, D., Ghahramani, Z., Bang, S.: Appearance-based Gender Classification with Gaussian Processes. Pattern Recognition Letters 27(6), 618–626 (2006)
6. Ueki, K., Komatsu, H., Imaizumi, S., Kaneko, K., Sekine, N., Katto, J., Kobayashi, T.: A Method of Gender Classification by Integrating Facial, Hairstyle, and Clothing Images. In: ICPR 2004, vol. 4 (2004)
7. Lapedriza, A., Masip, D., Vitria, J.: Are External Face Features Useful for Automatic Face Classification? In: CVPR 2005, vol. 3, pp. 151–157 (2005)
8. Su, Y., Shan., S., Chen, X., Gao, W.: Hierarchical Ensemble of Global and Local Classifiers for Face Recognition. In: ICCV 2007, pp. 1–8 (2007)
9. BrownU, E., Perrett, D.: What Gives a Face Its Gender? Perception 22, 829–840 (1993)
10. Cootes, T., Taylor, C., Cooper, D., Graham, J., et al.: Active Shape Models–Their Training and Application. Computer Vision and Image Understanding 61(1), 38–59 (1995)
11. Li, H., Jiang, T., Zhang, K.: Efficient and Robust Feature Extraction by Maximum Margin Criterion. IEEE Transactions on Neural Networks 17(1), 157–165 (2006)
12. Sugeno, M., Technology, T.I.: Theory of Fuzzy Integrals and Its Applications. PhD thesis, Tokyo Institute of Technology (1974)
13. Chang, S., Greenberg, S.: Application of Fuzzy-Integration-based Multipleinformation Aggregation in Automatic Speech Recognition. Intelligent Sensory Evaluation: Methodologies and Applications (2004)
14. Murofushi, T., Sugeno, M.: An Interpretation of Fuzzy Measures and the Choquet Integral as an Integral with Respect to a Fuzzy Measure. Fuzzy Sets and Systems 29(2), 201–227 (1989)
15. Wu, T., Lin, C., Weng, R.: Probability Estimates for Multi-Class Classification by Pairwise Coupling. The Journal of Machine Learning Research 5, 975–1005 (2004)
16. CAS-PEAL Face Database, http://www.jdl.ac.cn/peal/index.html
17. The POSTECH Face Database PF01, http://nova.postech.ac.kr/special/imdb/imdb.html
18. Yacoob, Y., Davis, L.S.: Detection and Analysis of Hair. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(7), 1164–1169 (2006)

A Hybrid Fuzzy Approach for Human Eye Gaze Pattern Recognition

Dingyun Zhu[1], B. Sumudu U. Mendis[1], Tom Gedeon[1], Akshay Asthana[2], and Roland Goecke[2]

[1] School of Computer Science
[2] School of Engineering,
The Australian National University, Acton, Canberra, ACT 0200, Australia
dingyun.zhu@anu.edu.au, sumudu.mendis@anu.edu.au,
tom.gedeon@anu.edu.au, aasthana@rsise.anu.edu.au,
roland.goecke@anu.edu.au.

Abstract. Face perception and text reading are two of the most developed visual perceptual skills in humans. Understanding which features in the respective visual patterns make them differ from each other is very important for us to investigate the correlation between human's visual behavior and cognitive processes. We introduce our fuzzy signatures with a Levenberg-Marquardt optimization method based hybrid approach for recognizing the different eye gaze patterns when a human is viewing faces or text documents. Our experimental results show the effectiveness of using this method for the real world case. A further comparison with Support Vector Machines (SVM) also demonstrates that by defining the classification process in a similar way to SVM, our hybrid approach is able to provide a comparable performance but with a more interpretable form of the learned structure.

1 Introduction

Human eyes and their movements are tightly coupled with human cognitive processes, which have been found to be very informative and valuable in various kinds of research areas. Furthermore, previous research has shown that human eye gaze patterns for observing different objects are also quite significant for the understanding of cognitive and decision-making processes.

We have been working on developing effective, efficient and robust approaches to generally provide a clear recognition or unambiguous interpretation of human eye gaze patterns in a variety of settings. In [4], we have successfully shown a sophisticated use of eye gaze information for inference of a user's intention in a game-like interactive task, which effectively eliminates the need of any physical control from the human's side, efficiently improving the communication between the user and the virtual agents.

In this paper, we introduce our hybrid fuzzy approach: hierarchical fuzzy signature construction with Levenberg-Marquardt learning of the generalized Weighted Relevance Aggregation Operator (WRAO) for modeling recognition of human eye gaze patterns between face scanning and text scanning.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 655–662, 2009.

2 Hierarchical Fuzzy Signatures

Hierarchical fuzzy signatures are fuzzy descriptors of real world objects. They represent the objects with the help of sets of available quantities which are arranged in a hierarchical structure expressing interconnectedness and sets of non-homogeneous qualitative measures, which are the interdependencies among the quantities of each set.

The fuzzy signature concept is an effective approach to solve the problem of rule explosion in traditional fuzzy inference systems: constructing characteristic fuzzy structures, modeling the complex structure of the data points (bottom up) in a hierarchical manner [6, 3, 11].

Fuzzy signatures start with a generalized representation of fuzzy sets which are regarded as *Vector Valued Fuzzy Sets (VVFS)* [6]. A Fuzzy Signature is a recursive version of VVFS where each vector can be another VVFS (called a branch) or an atomic value (called a leaf):

$$A : X \rightarrow [a_i]_{i=1}^{k} \tag{1}$$

$$where\ a_i = \begin{cases} [a_{ij}]_{j=1}^{k_i} & ;if\ branch \\ [0,1] & ;if\ leaf \end{cases} \tag{2}$$

Generally, fuzzy signatures result in a much reduced order of complexity, at the cost of slightly more complex aggregation techniques. Unlike conventional rule based hierarchical fuzzy systems, each branch in a fuzzy signature uses a different aggregation function to represent the importance of that branch to its parent, which is a final atomic value called "degree of match". Moreover, fuzzy signatures are different to conventional decision trees as well. They use a bottom up inference mechanism so that even with missing or noisy input data, this structure is still able to find a final result.

The fuzzy signature concept has been successfully applied to a number of applications, such as cooperative robot communication [14], personnel selection models [8], etc. Figure 1 is an example of a fuzzy signature structure which was constructed for a SARS pre-clinical diagnosis system [12].

$$\begin{bmatrix} Fever \begin{bmatrix} 8a.m. \\ 12a.m. \\ 4p.m. \\ 8p.m. \end{bmatrix} \\ BloodPressure \begin{bmatrix} Systolic \\ Diastolic \end{bmatrix} \\ Nausea \\ AbdominalPain \end{bmatrix}$$

Fig. 1. A Fuzzy Signature Example

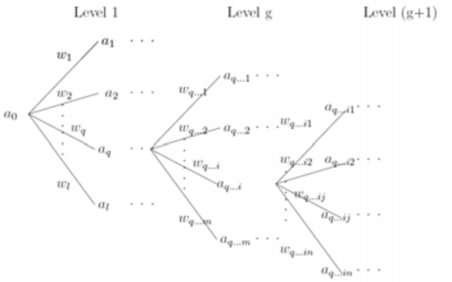


Fig. 2. An Arbitrary Fuzzy Signature Structure

3 Levenberg-Marquardt Learning of WRAO for Fuzzy Signatures

The Weighted Relevance Aggregation Operator (WRAO) [9] is derived from the generalization of the weights and aggregations in Weighted Relevance Aggregation (WRA), which introduces the weighted relevance of each branch to its higher branches of the fuzzy signature structure. In this way, WRAO is able to enhance the accuracy of the results of fuzzy signatures by allowing better adaptation to the meaning of the decision making process [10], and it can help to reduce the number of individual fuzzy signatures by absorbing more patterns into one structure.

The generalized Weighted Relevance Aggregation Operator (WRAO) of an arbitrary branch $a_{q...i}$ with n sub-branches, $a_{q...i1}$, $a_{q...i2}$,..., $a_{q...in} \in [0,1]$, and weighted relevancies, $w_{q...i1}$, $w_{q...i2}$,..., $w_{q...in} \in [0,1]$ (see Figure 2), for a fuzzy signature is a function g: $[0,1]^{2n} \rightarrow [0,1]$ such that,

$$a_{q...i} = \left[\frac{1}{n} \sum_{j=1}^{n} \left(a_{q...ij} \cdot w_{q...ij} \right)^{p_{q...i}} \right]^{\frac{1}{p_{q...i}}} \tag{3}$$

The Levenberg-Marquardt (LM) method is not only a major learning algorithm in neural network training functions, but also a widely used advanced approach that outperforms simple gradient descent and gradient methods for solving most of the optimization based problems. This algorithm is a Sum of Squared Error (SSE) based minimization method that is the function to be minimized is of the following special form [7]:

$$f(s) = \frac{1}{2} \sum_{i=1}^{n} \left(t_i - s_i \right)^2 = \frac{1}{2} \parallel \overline{t} - \overline{s} \parallel \tag{4}$$

where $\overline{t}$ stands for the target vector, $\overline{s}$ for the predicted output vector of the fuzzy signature, and $\|\|$ denotes the $2-norm$. Also, it will be assumed that there are m parameters to be learned and n records in the training data set, such that $n > m$. The next update of the LM is the following equation:

$$\underline{u}[k] = \underline{par}[k] - \underline{par}[k-1] \tag{5}$$

where the vector $\underline{par}[k]$ contains all the parameters to be learned, i.e. all the aggregation factors and weights of WRAO in the equation (3) for the kth iteration. Then the next update of $\underline{u}[k]$ is defined as:

$$\left(J^T[k] J[k] + \alpha I \right) \underline{u}[k] = -J^T[k] \underline{e}[k] \tag{6}$$

where J stands for the Jacobian matrix of the equation (4), I is the identity matrix of J, and α is a regularization parameter, which control both search direction and the magnitude of the next update $\underline{u}[k]$.

4 Eye Gaze Data Collection

An eye gaze data collecting experiment was conducted. Ten volunteers (Gender: 5 male, 5 female; Occupation: 2 academic staff, 6 postgraduates, 2 undergraduates) from the Australian National University community participated in the study.

Two sets of human face pictures, 20 in each, were selected for the face scanning experiment. Another 5 text only documents with different lengths (minimal half page, maximal one page) were also shown. In the experiment, all the face pictures and documents were demonstrated as full screen scenes on a monitor.

Every participant was firstly asked to view one set of human face pictures with about 5 seconds on each. The second stage of the experiment was to read the 5 text documents to determine which were the most important sentences in each one, no time restriction was imposed for the reading test so the participants could conduct the reading with their usual speed. In the ranking of sentences phase, only 1 participant ranked all the sentences, most participants ranked only 3 sentences so we conclude that our instructions were interpreted as a text scanning task. After that, the face scanning test was performed again with the other set of pictures as the last stage.

There was no time break between any two stages, all the eye movement data was collected by using a Seeingmachines eye-tracking system with FaceLAB software (Version 4.5, 2007) through the entire session of the experiment.

5 Fuzzy Signature Construction for Recognition of Eye Gaze Pattern

Since people tend to concentrate their gaze fixations onto the interesting and informative regions in the scene [13], we further filtered the original collected gaze points into fixations which offered a much easier and more interpretable form for the later data process. In addition, instead of considering all the fixations on each of the test cases (either face scanning or document scanning), we only use the first five fixations from every case. The reason for this is that it is possible to interpret a plausible eye gaze pattern from the early stage of face viewing (as early as the first five fixations) [1]. Moreover, the time period for scanning a document was obviously much longer than viewing a face in the data collecting experiment, so the decision to use only the first five fixations also maintains a more similar pattern for the future structure construction.

To construct the fuzzy signature structure for learning, it is necessary to figure out which essential feature in both of the possible patterns can show the difference for recognition. Figure 3 illustrates the first five fixations for two eye gaze patterns from face viewing as well as text scanning respectively. The two cases are obvious samples and this is actually not the usual source in all the data records we collected in the experiment.

From these two cases, we can easily find the most obvious difference between them is in the geometrical shapes, which shows that compared with face

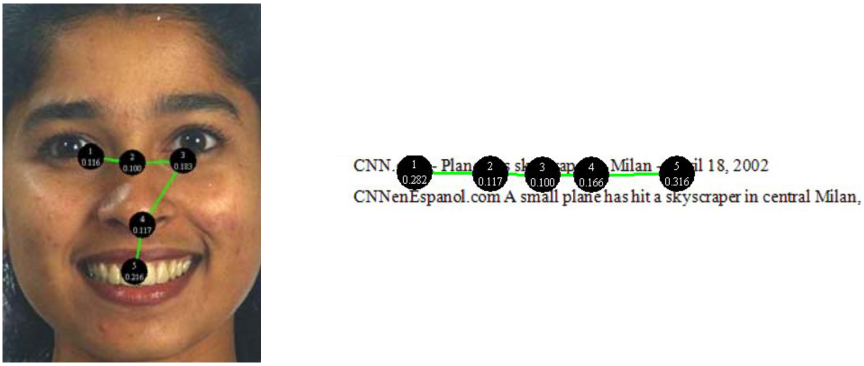

1
0.116
2
0.100
3
0.183
4
0.117
5
0.216
CNN. 1 0.282 - Plan 2 0.117 s 3 0.100 sk ap 4 0.166 Milan 5 0.316 il 18, 2002
CNNenEspanol.com A small plane has hit a skyscraper in central Milan,

Fig. 3. Two Samples of First Five Fixations Only Eye Gaze Patterns

scanning, participants' gaze fixation locations for text scanning follow a very clear horizontal pattern. On the other hand, although it is still difficult to address a common gaze pattern for the face scanning, the plausible pattern has a much more complicated geometrical shape than the simple form from text scanning, because the informative regions (eyes, nose, mouth and cheeks, etc) in which an observer is interested in a face are not aligned horizontally as are the sentences in a document.

According to the above point, we can further discover that the actual feature in both of the patterns rests on the vertical difference between two fixations which are adjacent on time. Consequently, the constructed fuzzy signatures for the recognition of two patterns can be formed to the structure in Figure 4.

The leaves of each sub-signature in the structure represent the fuzzy value calculated by using the Fuzzy C Mean (FCM) clustering method based on the vertical difference between two adjacent fixations.

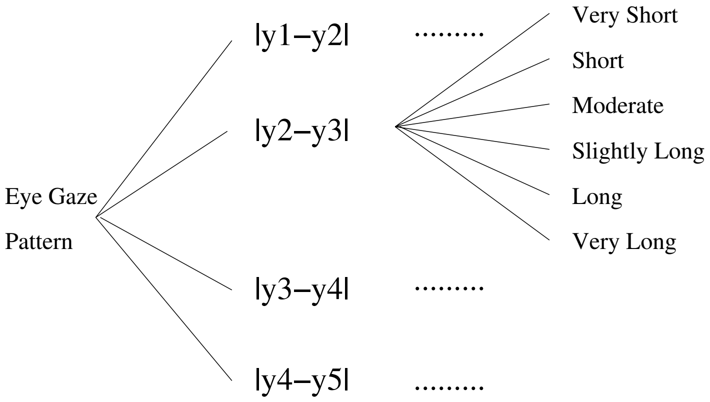


Fig. 4. Fuzzy Signature Structure for Eye Gaze Pattern

Table 1. Fuzzy Signatures Results for Eye gaze Pattern Recognition

Experiment	Mean Squared Error (MSE)	Classification Error (CLE)
Training	0.0605	19.77%
Test	0.0641	20.00%

6 Evaluation and Comparison

According to the constructed fuzzy signature structure for the recognition of these two different patterns, we performed experiments to learn the weights and aggregations by applying the Levenberg-Marquardt optimization method as we explained in the previous section. The following table shows the results of Mean Squared Error (MSE) and Classification Error (CLE) for learning the weights of WRAO for the training and test experiments respectively.

We need to clarify that the way we calculate the Classification Error (CLE) for the fuzzy signature structure is actually based on the degree of difference (d) between the predicted value and the initial desired value. In our eye gaze pattern case, we set three classes according to the degree of difference: *Good* ($|d| \leq 0.2$, not an error), *Bad* ($0.2 < |d| \leq 0.5$, count 0.5 error) and *Very Bad* ($|d| > 0.5$, count 1 error). So the final classification error rate would be the sum of all the error numbers divided by the total number of records in the data set.

Table 1 shows that our hybrid fuzzy approach can perform around 80% accurate predictions for the recognition of different eye gaze patterns between face and text scanning.

Furthermore, in order to have a performance comparison, the Support Vector Machines (SVM) [5] based classifier was chosen to run through the same eye gaze fixation data set. Since the classification problem here is only for the recognition between two different eye gaze patterns, we constructed a simple SVM based classifier using a linear kernel to classify the data of vertical difference between two adjacent fixations as we used in the previous experiment. For our fuzzy signature structure, we also reduced the number of classes from previous three (*Good, Bad and Very Bad*) to two (*Good* $|d| \leq 0.5$ and *Bad* $|d| > 0.5$). Table 2 demonstrates the results of the experiments between fuzzy signatures and SVM.

From the above results we can see for this particular pattern recognition problem, the simple SVM based classifier constructed by using a linear kernel gives highly accurate classification results. Comparatively, by reducing the number of classes for the fuzzy signatures from previous three (*Good, Bad and Very Bad*) to two (*Good* $|d| \leq 0.5$ and *Bad* $|d| > 0.5$), the classification error rate reduces and is comparable to that of SVM.

Table 2. CLE Comparison Between Fuzzy Signatures and SVM

Experiment	FS (3 classes)	FS (2 classes)	SVM (2 classes)
Training	19.77%	3.04%	1.18%
Test	20.00%	5.00%	4.55%

Beyond the results comparison, we also find that the way fuzzy signatures model the classification problem is different from the way SVM models a classification problem. SVM approaches the classification problem through the concept of *margin*, that is equal to the smallest distance between the decision boundary and any of the samples [2]. It computes the maximum margin hyperplane that best defines the decision boundary. The samples that are closest to the decision boundary lie on this maximum margin hyperplane and are know as support vectors. Any other sample point in the data set plays no role in the classification problem and is discarded. Once the decision boundary is known, the new data points can be classified according to which side of the hyperplane it lies. On the other hand, the construction of fuzzy signature is based on the expression of the domain knowledge. Further, both the weight and aggregation learning processes for WRAO offer us a clear view of what exactly produces the results inside the structure, for instance, which aggregation functions are learned for each branch. In addition, the value generated after every aggregation actually represents the degree of match as the importance of the current branch to its parent, which is also useful to discover which sub-branch makes the most contribution and which are not significant factors from the domain for the problem modeling. So fuzzy signatures provide a more interpretable expression of how well the output of the structure approaches the target value or classification.

7 Conclusion

A fuzzy signature with a Levenberg-Marquardt learning based hybrid approach has been introduced for modeling a real world study: recognizing human eye gaze patterns to distinguish between face scanning and text scanning. The constructed structure shows significant performance for the recognition in the experiment results.

From the further discussion of performance comparison with linear SVM, we suggest that our structure is capable of producing a comparable result for the classification, but as an effective approach, it can provide a more interpretable representation signature pattern of a real world object from its construction as well as the learning process.

References

[1] Althoff, R.R., Cohen, N.J.: Eye-movement Based Memory Effect: A Reprocessing Effect in Face Perception. Journal of Experimental Psychology: Learning, Memory, and Cognition 25, 997–1010 (1999)

[2] Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)

[3] Gedeon, T.D., Koczy, L.T., Wong, K.W., Liu, P.: Effective Fuzzy Systems for Complex Structured Data. In: Proceedings of IASTED International Conference Control and Applications (CA 2001), Banff, Canada, pp. 184–187 (2001)

[4] Gedeon, T.D., Zhu, D., Mendis, B.S.U.: Eye Gaze Assistance for a Game-like Interactive Task. International Journal of Computer Games Technology 2008, Article ID 623752(2008)

[5] Hsu, C., Chang, C., Lin, C.: A Practical Guide to Support Vector Classification. Technical Report, Department of Computer Science, National Taiwan University (2005)

[6] Koczy, L.T., Vamos, T., Biro, G.: Fuzzy Signatures. In: Proceedings of the 4th Meeting of the Euro Working Group on Fuzzy Sets and the 2nd International Conference on Soft and Intelligent Computing (EUROPUSE-SIC 1999), Budapest, Hungary, pp. 210–217 (1999)

[7] Marquardt, D.: An Algorithm for Least Squares Estimation of Nonlinear Parameters. Journal of the Society for Industrial and Applied Mathematics 11(2), 431–441 (1963)

[8] Mendis, B.S.U., Gedeon, T.D.: A Comparison: Fuzzy Signatures and Choquet Integral. In: Zurada, J.M., Yen, G.G., Wang, J. (eds.) WCCI 2008. LNCS, vol. 5050, pp. 1464–1471. Springer, Heidelberg (2008)

[9] Mendis, B.S.U., Gedeon, T.D., Botzheim, J., Koczy, L.T.: Generalized Weighted Relevance Aggregation Operators for Hierarchical Fuzzy Signatures. In: Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA 2006), Jointly with the International Conference on Intelligent Agents, Web Technologies and International Commerce (IAWTIC 2006), Sydney, Australia, pp. 198–203 (2006)

[10] Mendis, B.S.U., Gedeon, T.D., Koczy, L.T.: Learning Generalized Weighted Relevance Aggregation Operators Using Levenberg-Marquardt Method. In: Proceedings of the 6th International Conference on Hybrid Intelligent System (HIS 2006) and 4th Conference on Neuro-Computing and Evolving Intelligence (NCEI 2006), New Zealand, pp. 34–39 (2006)

[11] Vamos, T., Koczy, L.T., Biro, G.: Fuzzy Signatures in Data Mining. In: Proceedings of Joint 9th IFSA World Congress and 20th NAFIPS International Conference, Vancouver, Canada, vol. 5, pp. 2841–2846 (2001)

[12] Wong, K.W., Gedeon, T.D., Koczy, L.T.: Construction of Fuzzy Signature from Data: An Example of SARS Pre-clinical Diagnosis System. In: Processings of IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2004), Budapest, Hungary, pp. 1649–1654 (2004)

[13] Yarbus, A.: Eye Movements and Vision. Plenum Press, New York (1967)

[14] Zhu, D., Gedeon, T.: Fuzzy Logic for Cooperative Robot Communication. In: Nguyen, N.T., Jo, G.S., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2008. LNCS, vol. 4953, pp. 401–410. Springer, Heidelberg (2008)

Interactive Trouble Condition Sign Discovery for Hydroelectric Power Plants

Takashi Onoda[1], Norihiko Ito[1], and Hironobu Yamasaki[2]

[1] System Engineering System Laboratory,
Central Research Institute of Electric Power Industry,
2-11-1, Iwado Kita, Komae-shi, Tokyo 201-8511 Japan
{onoda,norihiko}@criepi.denken.or.jp
[2] The Power System Engineering Department,
Kyushu Electric Power Co.,Inc.,
2-1-82, Watanabe-Dori, Chuo-ku, Fukuoka 810-8720 Japan
Hironobu_Yamasaki@kyuden.co.jp

Abstract. Kyushu Electric Power Co.,Inc. collects different sensor data and weather information (hereafter, operation data) to maintain the safety of hydroelectric power plants while the plants are running. It is very rare to occur trouble condition in the plants. And it is hard to construct an experimental power generation plant for collecting the trouble condition data. Because its cost is too high. In this situation, we have to find trouble condition sign. In this paper, we consider that the rise inclination of *special unusual condition data* gives trouble condition sign. And we propose a trouble condition sign discovery method for hydroelectric power plants by using a one class support vector machine and a normal support vector machine. This paper shows the proposed method is useful method as a method of risk management for hydroelectric power plants.

Keywords: Data Mining, Trouble Condition Detection, Support Vector Machine, Hydroelectric Power Plant.

1 Introduction

Recently, electric power companies have begun to try to shift a Time Based Maintenance (hereafter, we use TBM) to a Condition Based Maintenance (hereafter, we use CBM) for electric equipment management to realize an efficient maintenance and reduce the cost of maintenance[1,2]. TBM is to check and change the equipment based on the guaranteed term recommended by makers. And CBM is to check, repair and change the equipment based on the condition of equipment[3]. The condition consists of the present state of equipment, the operation term of equipment, the load in an operation, and etc[3]. Therefore, this CBM is a kind of risk management for electric power companies' management.

It is important for electric power companies to collect the data of equipment to realize the CBM. Especially, it is necessary to collect and analyze past trouble condition data for discovering a trouble condition sign of power equipment[4,5].

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 663–670, 2009.

Table 1. Outline of a targeted Hydroelectric Power Plant

Generated Output		18,000kW
Working Water		45.00m^3/s
Effective Head		46.30m
Turbine Type		Vertical Shaft Francis Turbine
Rated Revolutions Per Minute		240rpm
Bearing Type	Upper Bearing	Oil Self Contained Type Segment Bearing(Natural Cooling)
	Bottom Bearing	Oil Self Contained Type Segment Bearing(Natural Cooling)
	Turbine Bearing	Oil Self Contained Type Cylindrical Bearing(Natural Cooling)
	Thrust Bearing	Oil Self Contained Type Pivot Spring Bearing(Natural Cooling)
Operation Pattern		Process Control Operation (An operation pattern is generated at everyday.)

For instance, power companies want to develop a trouble condition sign discovery method from operation data, which consists of sensor information of the hydroelectric power plant and weather information. However, it is hard to collect the trouble condition data in the plant in Japan. Because power companies have changed the power plant equipment in certified term by makers for the conventional maintenance and it is rare to occur a trouble condition. Instead of collecting the trouble condition data from real hydroelectric power plants, we can consider that we collect the trouble data from an experimental hydroelectric power plant. But it is hard to construct it, because its cost is too high. Therefore, it is impossible to acquire the operation data in the trouble condition.

Kyushu Electric Power Co.,Inc. and Central Research Institute of Power Industry are researching the discovery of the trouble condition signs based on an unusual condition detection for the bearing vibration using the operation data, now. In our research, we consider that the rise inclination of *special unusual condition data* gives trouble condition sign, because we can measure the regular condition data only from hydroelectric power plants and both the trouble condition and the *special unusual condition* are very rare cases. The *special unusual condition data* will be described in the section 3.

In this paper, we describe the operation data for a hydroelectric power plat briefly in the next section. In the third section, we briefly explain our proposed method to discover a trouble condition sign for bearing vibration. The experimental results are shown in the forth section. Finally, we conclude our research and describe our future work.

2 Measurement Data

Table 1 shows the outline of a hydroelectric power plant. The hydroelectric power plant has various sensors to measure data related to bearing vibration. In this paper, the operation data is collected from the hydroelectric power plant and analyzed by our proposed method. The operation data, which is related to bearing vibration, is collected from April 1, 2006 to January 31, 2008.

One operation data has been composed of the sensor and weather data on 44 measurement items for five seconds the measurement interval. All operation data is regular condition data and does not include an trouble condition data.

3 Trouble Condition Sign Detection Approach

In this section, we describe the outline of our meted of discovery of the trouble condition sign using the *special unusual condition data.*

Generally, the discovery of an trouble condition sign is to detect a peculiar case, which appears only before a trouble condition, by comparing between regular condition data and trouble condition data. It is a fact that there is little data of a trouble condition data in the electric power plants, because the electric power plants is designed with the high safety factor. Currently, our bearing vibration data of the hydroelectric power plant also does not have trouble condition data. Therefore, it is impossible to discover the peculiar case before trouble condition data happen. Then, we think the relation between the peculiar condition before a trouble condition data happen (hereafter, we call it the trouble condition sign) and *special unusual condition data* as the following relation.

$$\begin{aligned} \textit{The trouble condition sign} \approx\ & \textit{The strongly rise inclination} \\ & \textit{of special unusual condition data.} \end{aligned}$$

It is possible to change the discovery of the trouble condition sign to the detection of the *special unusual condition data* in the regular condition data under this assumption. In other words, we supposes that the *special unusual condition data* has low probability of existing in the regular condition data has high probability of trouble condition sign.

Our proposed trouble condition sign discovery method integrates the detection method of *special unusual condition data* and the trace method of the trend of generating *special unusual condition data.* The figure 1 shows an image of the trouble condition sign discovery for the condition based maintenance of hydroelectric power plants. Our proposed trouble condition sign discovery method is an interactive method. The system consists of two approaches mainly. One is an selection method of the *special unusual condition data*, which relate to trouble condition sign. The method consists of two phases. One phase is the unusual condition data detection phase based on a One-Class Support Vector Machine(hereafter, we call 1-SVM) and the next phase is the *special unusual condition data* selection. The detected unusual condition data by 1-SVM may include sensor fault data and missing value data and so on. These data are definitely scrap condition data. In the other phase, human experts can detect these scrap condition data by using their expertise and operation reports etc, and select the data related to trouble condition sign in the detected unusual condition data by 1-SVM. This selected unusual condition data is defined as *the special unusual condition data.* And the other approach is the generation trend tracing method based on a normal Support Vector Machine(hereafter, we call SVM). The experts can teach the *special condition data* to the computer. After that, the computer has the usual condition data and the *special condition data.* The computer can generate an optimal hyper plane, which can classify the two classes, by using a SVM. The hyper plane classifies unseen data and finds some data which are similar to the *special selected unusual condition data.* Therefore, the computer can trace the trend of the generation of the *special unusual condition data.*

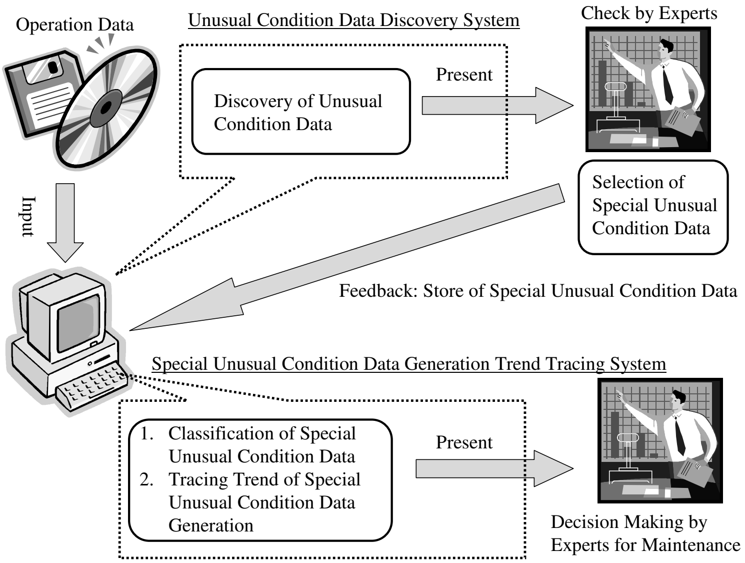

Operation Data
Unusual Condition Data Discovery System
Check by Experts
Discovery of Unusual Condition Data
Present
Input
Selection of Special Unusual Condition Data
Feedback: Store of Special Unusual Condition Data
Special Unusual Condition Data Generation Trend Tracing System
1. Classification of Special Unusual Condition Data
2. Tracing Trend of Special Unusual Condition Data Generation
Present
Decision Making by Experts for Maintenance

Fig. 1. Image of Trouble Condition Sign Discovery

In our research, we used the LIBSVM. This is an integrated tool for support vector classification and regression which can handle One Class SVM using the Schölkopf etc algorithms. The LIBSVM is available at *http://www.csie.ntu.edu.tw /˜cjlin/libsvm*.

4 Trouble Condition Sign Detection Experiment

In this section, we describe our experiment by using the operation data, which is explained in section 2. Especially, we briefly introduce our experimental setup, our experimental results and the evaluation.

4.1 Experimental Setup

Our experiment analyzed the operation data, which is explained in section 2. The operation data is composed of 44 measurement items. However, in order to detect the unusual condition data of the bearing, we extracted the related measurement items to the bearing vibration from all measurement items. Therefore, 13 measurement items were selected by the expertise of the bearing vibration of the experts to analyze the unusual condition data for the starting condition and the parallel off condition except for the generated output. The two condition do not generate output. 14 measurement items were selected for the parallel condition. Table 2 shows these selected 14 measurement items.

The power generator operation consists of the starting condition, the parallel condition, the parallel off condition and the stopping condition. The starting

Table 2. Measurement Items

Measurement Items for Detection Analysis	
A. Generated Output(MW)	B. Revolutions Per Minute
C. Upper Bearing Oil Tempe. - Oil Cooler Inlet Air Tempe.(℃)	D. Turbine Bearing Oil Tempe.(℃)
E. Thrust Bearing Tempe.(℃)	F. Bottom Oil Tank Oil Tempe.(℃)
G. Bottom Bearing Inlet Air Tempe.(℃)	H. Turbine Shaft Vibration(X axis)(μm)
I. Upper Bearing Vibration(horizon)(μm)	J. Upper Bearing Vibration(perpendicular)(μm)
K. Bottom Bearing Vibration(horizon)(μm)	L. Bottom Bearing Vibration(perpendicular)(μm)
M. Turbine Shaft Vibration(horizon)(μm)	N. Turbine Shaft Vibration(perpendicular)(μm)

condition data and the parallel off condition data are very few in our data set relatively. The parallel operation condition data are very large. If we analyze the all operation data to detect the unusual condition data, the detected unusual condition data are the starting condition data or the parallel off condition data. This is not good situation for our analysis. Therefore, the all operation data is divided into the following four conditions by expertise of experts.

Starting condition:
Generator Voltage(V-W) $<$ 10kV and Guide Vane Opening $\geq$ 10% and Revolutions Per Minute $\geq$ 200 rpm.

Parallel operation condition:
Generator Voltage(V-W) $\geq$ 10kV and Revolutions Per Minute $\geq$ 200 rpm.

Parallel off condition:
Generator Voltage(V-W) $<$ 10kV and Guide Vane Opening $<$ 10% and Revolutions Per Minute $\geq$ 200 rpm.

Stopping condition:
Otherwise.

These conditions are defined by the expertise of the experts. The data were measured from April 1 in 2006 to January 31 in 2008.

In the stopping condition, the bearing does not rotate. This group data were omitted from the analyzed data. In order to ignore the different measurement units, the operation data is normalized by the the following equation at each measurement item.

$$\text{value} = -\frac{\text{actual value} - \text{min. value}}{\text{max. value} - \text{min. value}} + 1$$

For hydroelectric power plant, high value of each sensor denotes unsafety. Therefore, our method adopted the normalization.

In order to evaluate our proposed method for trouble condition sign discovery, we need two data sets. One data set is for the detection of unusual condition data and the selection of *special unusual condition data*. The other is for tracing the trend of the *special unusual condition data* generation. In this research, the operation data is divided two data sets. One data set is from April 1 in 2006 to March 31 in 2007 for the detection and the selection. The other data set is April 1 in 2007 to January 31 in 2008 for the tracing the trend.

4.2 Experimental Results

Experimental Results for Unusual Condition Data Detection. The unusual condition data were detected in each condition data of the starting condition, the parallel condition and the parallel off condition by applying 1-SVM. And the trend of *special unusual condition data* generation is traced in each condition. In order to detect unusual condition data, this experiment used the operation data measured from April 1 in 2006 to March 31 in 2007. In our experiment, about 15 unusual condition data were detected from each condition data of the starting condition, the parallel condition and the parallel off condition. The number of the detected unusual condition data is determined by the expertise of human experts. It is easy for human experts to explain that the data, which are more than 15, are usual data. Therefore, we set the number of the detected unusual data to 15.

The number of the starting condition data is 6,279. The parameter ν of 1-SVM was set to 0.0024 to detect the unusual condition data from the starting condition data. This parameter denotes a detection rate. The number of the parallel condition data is 1,285,023. The parameter ν of 1-SVM was set to 1.17×10^{-5} to detect the unusual condition data from the parallel condition data. The number of the parallel off condition data is 627. The parameter ν of 1-SVM was set to 0.24 to detect the unusual condition data from the starting condition data. After all, our proposed method could detect 16 unusual condition data from each condition data of the starting condition, the parallel condition and the parallel off condition. These detected unusual condition data were checked by human experts. The human experts selected the all detected unusual condition data as *special unusual condition data*, which may indicate the trouble condition sign.

Experimental Results for Trouble Condition Sign Discovery. SVM generated a discriminate function, which can classify between the *special unusual condition data* and usual condition data in each operation condition. The operation data, which were measured from April 1 in 2007 to January 31 in 2008, are input into the discriminate function. And our proposed method computed the generation rate of the *special unusual condition data* from April 1 in 2007 to January 31 in 2008. The table 3 shows the generation rate of the *special unusual condition data* in the starting condition data for each month. The table 4 shows

Table 3. The generation rate of the *special unusual condition data* in the starting condition data

Month	April	May	June	July	August	September	October	November	December	January	Average
No. of generation data	2	8	9	33	29	9	1	0	0	0	9.1
No. of data	1203	503	630	106	125	330	576	682	612	657	542.40
Generation rate(%)	0.166	1.59	1.43	31.13	23.2	2.73	0.17	0.00	0.00	0.00	1.68

Table 4. The generation rate of the *special unusual condition data* in the parallel condition data

Month	April	May	June	July	August	September
No. of generation data	1	0	2	197	5	0
No. of data	86485	154839	113963	168666	168599	129222
Generation rate(%)	0.0012	0.0000	0.0018	0.1168	0.0030	0.0000
Month	October	November	December	January		Average
No. of generation data	0	0	0	0		20.50
No. of data	64788	47352	22438	51209		100756.10
Generation rate(%)	0.0000	0.0000	0.0000	0.0000		0.0203

Table 5. The generation rate of the *special unusual condition data* in the parallel off condition data

Month	April	May	June	July	August	September
No. of generation data	7	5	12	4	3	4
No. of data	128	54	68	10	15	31
Generation rate(%)	5.47	9.26	17.65	40.00	20.00	12.90
Month	October	November	December	January		Average
No. of generation data	4	1	0	1		4.10
No. of data	60	75	69	61		57.10
Generation rate(%)	6.67	1.33	0.00	1.64		7.18

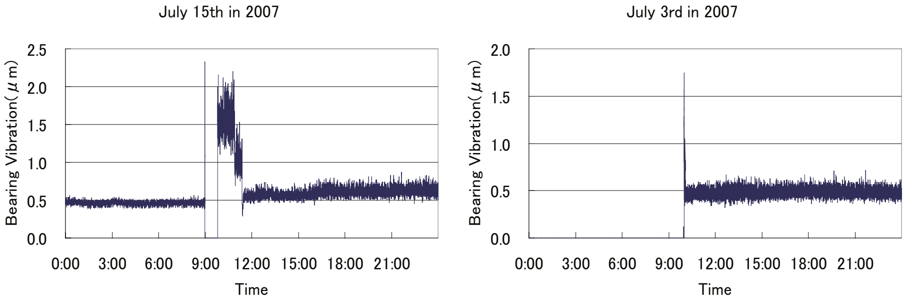


Fig. 2. The time series data of bearing vibration

the generation rate of the *special unusual condition data* in the parallel condition data for each month. The table 5 shows the generation rate of the *special unusual condition data* in the parallel off condition data for each month.

From tables 3, 4, 5, we can find a high value of the generation rate in July for each operation condition. This value of the generation rate is much higher than the value of the detection rate. Except for July, the generation rates is similar to the detection rate for each month. The *special unusual condition data* in July were measured in July 15th. Figure 2 shows time series data of the bearing vibration in July 15th and July 3rd. In the figure, In the figure, the left side figure shows time series data of the bearing vibration in July 15th, and the right side figure shows time series data of the bearing vibration in July 3rd and the regular bearing vibration data.

We can find high values of bearing vibration from 10 a.m. to 11 a.m. in the left side figure. The bearing vibration data were classified to the *special unusual condition data*. The left side figure is different from the right side figure. The right side figure shows the regular bearing vibration data in the parallel condition. The left side figure is not regular. We could investigate that this operation kept 20-30 % of generated power in operation records. This operation is very strange and very rare case. Therefore, our proposed method can discover strange and rare data, which may indicate the trouble condition.

5 Conclusion

In this paper, we proposed the interactive trouble condition sign discovery method based on the unusual condition detection. The proposed method detects the unusual condition data of bearing vibration in hydroelectric power plants base on the 1-SVM, primarily. Then our proposed method selects the *special unusual condition data*, which may indicate the trouble condition sign, by using expertise of human experts. And our method traces the trend of the *special unusual condition data* generation in unseen data. If the generation rate of the *special unusual condition data* is increasing strongly, a trouble condition is coming, our method think so.

In this paper, we applied our method to the operation data of a real hydroelectric power plant. And our experiment showed that our method can discovery strange and rare data, which may indicate a trouble condition sign.

References

1. Yamana, M., Murata, H., Onoda, T., Oohashi, T., Kato, S.: Comparison of pattern classification methods in system for crossarm reuse judgement on the basis of rust images. In: Proceedings of Artificial Intelligence and Applications 2005, pp. 439–444 (2005)
2. Jardine, A.K.S.: Repairable system reliability: Recent developments in CBM optimization. In: 19th International Congress and Exhibition on Condition Monitoring and Diagnostic Engineering Management (COMADEM), Luleä, Sweden, June 13-15 (2006)
3. Tsang, A.H.C., Yeung, W.K., Jardine, A.K.S., Leung, P.K.: Data management for CBM optimization. Journal of Quality in Maintenance Engineering 12, 37–51 (2006)
4. Lin, D., Banjevic, D., Jardine, A.K.S.: Using principal components in a proportional hazards model with applications in condition-based maintenance. The Journal of the Operational Research Society 57, 910–919 (2006)
5. Zuashkiani, A., Banjevic, D., Jardine, A.K.S.: Incorporating expert knowledge when estimating parameters of the proportional hazards model. In: Proceedings of Reliability and Maintainability Symposium, Newport Beach, CA, January 23-26 (2006)
6. Schölkopf, B., Smola, A., Williamson, R., Bartlett, P.: New support vector algorithms. Neural Computation 12, 1083–1121 (2000)
7. Cortes, C., Vapnik, V.: Support Vector Networks. Machine Learning 20, 273–297 (1995)
8. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, Heidelberg (1995)
9. Vapnik, V.N.: Statistical Learning Theory. Wiley, New York (1998)

An Asbestos Counting Method from Microscope Images of Building Materials Using Summation Kernel of Color and Shape

Atsuo Nomoto, Kazuhiro Hotta, and Haruhisa Takahashi

The University of Electro-Communications,
1-5-1 Chofugaoka Chofu-shi Tokyo, 182-8585, Japan
{nomoto,hotta,takahasi}@ice.uec.ac.jp

Abstract. In this paper, an asbestos counting method from microscope images of building materials is proposed. Since asbestos particles have unique color and shape, we use color and shape features for detecting and counting asbestos by computer. To classify asbestos and other particles, the Support Vector Machine (SVM) is used. When one kernel is applied to a feature vector which consists of color and shape, the similarity of each feature is not used effectively. Thus, kernels are applied to color and shape independently, and the summation kernel of color and shape is used. We confirm that the accuracy of asbestos detection is improved by using the summation kernel.

1 Introduction

In the last decade, health damages by asbestos become a big problem in the world. Asbestos is a fibrous mineral. It is nonflammable, durable and inexpensive. Therefore, it is especially suitable as insulation and is easy to handle. From such reasons, asbestos was widely used as building materials in Japan after the high-growth period of the 1970s. However, the use of asbestos has been banned or limited worldwide since the late 1980s, because it was discovered to cause cancer.

However, asbestos remains in buildings built in the past yet. We need to check whether asbestos are used or not when buildings are demolished. Now, a human investigator checks the specimen of building materials by watching the microscope images. This method is not only inefficient, but human burden is large. Therefore, we desire the system which carries out automatic counting of asbestos particles.

The asbestos analysis by human investigators is called as a disperse dyeing method. In disperse dyeing method, an investigator prepares three specimens from one sample, and counts 1000 particles from by each specimen. When more than four asbestos are contained in 3000 particles, it is judged as hazardous. There is no hazardous in asbestos particles itself and only asbestos whose shape is needle-like (aspect ratio is more than 3:1) is hazardous. The needle-like shape is a criterion for counting asbestos particles. Asbestos particles have another property. That is the polarization property in which asbestos particles emit

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 671–678, 2009.

particular color. The investigators count asbestos using both color and shape properties. Therefore, it is considered that image recognition is most appropriate method for detecting and counting asbestos particles automatically by computer.

In this paper, we propose the automatic asbestos detection and counting method by computer. By using image recognition technique based on statistical methods, we achieved accurate detection results. Since the asbestos detection from microscope images is the two classification problem such as face or pedestrian detection [6,7,8,9], we use SVM [2] which is a binary classifier with high generalization ability. Figure 1 shows the overview of the proposed method. To train the detector, we use the many asbestos and non-asbestos images. Since the size of asbestos is not constant, we judge whether a local region with M×M pixels contains asbestos or not. From the classification results of local regions, we detect and count asbestos. Since the asbestos particles have characteristic shape and color, we use those features. However, when one kernel is applied to a feature vector which consists of color and shape, the similarity of each feature is not used effectively. Thus, we apply kernels to color and shape feature independently, and the outputs of kernels are integrated by summation. We show that this kernel improves the accuracy. In detection and counting process, the detector is adopted to every local region in a test image, and a Map image in which white pixels show high SVM output and black shows low SVM output is obtained. The Map image is binarized, and asbestos particles are counted by labeling.

This paper is organized as follow. In section 2, we describe the color and shape feature for counting asbestos particles with specific polarization. Section 3 describes the asbestos detection method which is based on SVM with the summation kernels [1]. The asbestos counting method is explained in section 4. Experimental results are shown in section 5, followed by the conclusions in section 6.

2 Features for Asbestos Detection

Since asbestos particles emit a particular color with specific polarization. Therefore, it is considered that color information is important. However, the direction or position of asbestos particles in a local region are not constant. In order to be robust to these changes, we use histogram of RGB color information. However, when only color feature is used, asbestos particles are not detected accurately in experiments because there is the case that non-asbestos particles have the color histogram similar to asbestos's histogram. Therefore, we focus the needle-like shape of asbestos particles. However, the position of asbestos in a local region is not constant. Thus, we calculate an average of edge in a local region. The edges are calculated by Sobel filters of four directions which are shown in Figure 2. Since the direction of asbestos is not constant, the variance of four directional edges may be more effective, and we also use the variance. These features show rough shapes overall. In this paper, we use both color histogram and shape feature for detection.

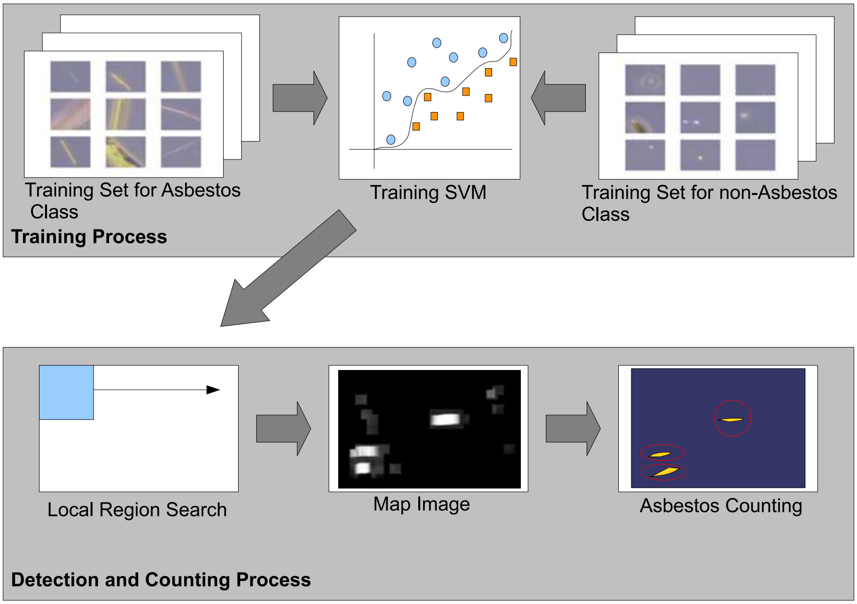

Training Set for Asbestos Class
Training SVM
Training Set for non-Asbestos Class
Training Process
Local Region Search
Map Image
Asbestos Counting
Detection and Counting Process

Fig. 1. Overview of asbestos counting system

3 Asbestos Detection

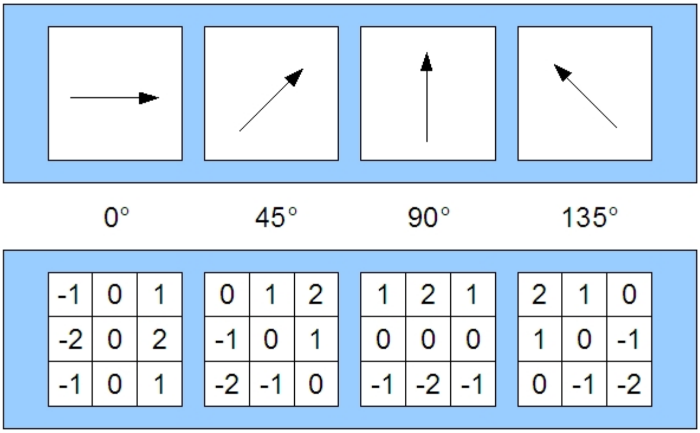
In SVM, we can adopt one kernel to a feature vector which consists of color and shape. However, in that case, the similarity of each feature is not used effectively. To avoid this, we use the summation kernel [1,3] of color and shape. By assigning kernels to each feature and calculating the summation of each output, we obtain a new kernel function which uses the similarity of each feature effectively. The new kernel function K_{new} is defined as

$$K_{new}(\boldsymbol{x}, \boldsymbol{z}) = K_c(\boldsymbol{x}_c, \boldsymbol{z}_c) + K_s(\boldsymbol{x}_s, \boldsymbol{z}_s), \tag{1}$$

where K_c and K_s are the kernels for color and shape, $\boldsymbol{x}$ and $\boldsymbol{z}$ are the input vectors. To integrate both kernels fairly, the normalized polynomial kernel [4] which normalizes the norm of mapped feature is used. The kernel is defined as

$$\begin{aligned} K(\boldsymbol{x}, \boldsymbol{z}) &= \frac{\phi^T(\boldsymbol{x})\phi(\boldsymbol{z})}{\|\phi(\boldsymbol{x})\|\|\phi(\boldsymbol{z})\|} \\ &= \frac{(1+ < \boldsymbol{x}, \boldsymbol{z} >)^d}{\sqrt{(1+ < \boldsymbol{x}, \boldsymbol{x} >)^d}\sqrt{(1+ < \boldsymbol{z}, \boldsymbol{z} >)^d}}. \end{aligned} \tag{2}$$

By normalizing the output of standard polynomial kernel, the kernel output is between 0 and 1. In [1], all kernels are integrated with equal weights. However,



-1	0	1
-2	0	2
-1	0	1

0	1	2
-1	0	1
-2	-1	0

1	2	1
0	0	0
-1	-2	-1

2	1	0
1	0	-1
0	-1	-2

Fig. 2. Direction of Sobel filter. The 1st row shows the 4 different directions, and the 2nd row shows the operators of Sobel filter corresponding to the direction. The shape feature is calculated using the operators.

the summation kernel with non-negative weight also satisfies Mercer's theorem [5]. In this paper, we use the weight to integrate the two kernels.

$$K_{new}(\boldsymbol{x}, \boldsymbol{z}) = \alpha K_c(\boldsymbol{x}_c, \boldsymbol{z}_c) + (1 - \alpha) K_s(\boldsymbol{x}_s, \boldsymbol{z}_s), \tag{3}$$

where α is a constant between 0 and 1. In the following experiments, we set the parameters as $d = 3$ and $\alpha = 0.6$ by preliminary experiment. An asbestos detector based on SVM with the weighted summation kernel is trained using many asbestos and non-asbestos images.

4 Asbestos Counting

In counting process, our goal is to count the asbestos particles from the Map image. The Map image is a grayscale image whose intensity is the output of the asbestos detector based on SVM. In the Map image, the high pixel value means that the probability of asbestos particle's existence is high. Figure 3 shows how to obtain the Map image. We assign the output of the asbestos detector to the center pixel in a local region. This process is repeated over a whole image. A pixel value of the Map image shows the degree of asbestos particle's existence. Therefore, we can divide the Map image into the two regions by a threshold, the regions which probably contain an asbestos particle and the regions which do not contain. Then we obtain the binary image from the Map image. The threshold for the binarization is set to zero which corresponds to the boundary between positives and negatives in SVM. We count asbestos particles by labeling in the binarized image.

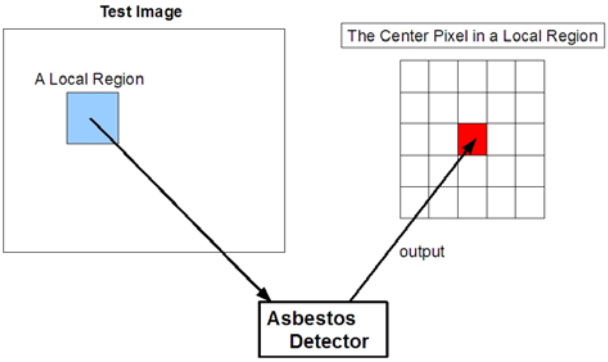


Fig. 3. How to obtain the Map image. In every local region, we assign the output of asbestos detector to the center pixel of the local region.

5 Experiments

This section shows experimental results. First, we describes image dataset in section 5.1. Evaluation results are shown in section 5.2. The effectiveness of the summation kernel is demonstrated by the comparison with a kernel of only color feature and a kernel of joint feature of color and shape.

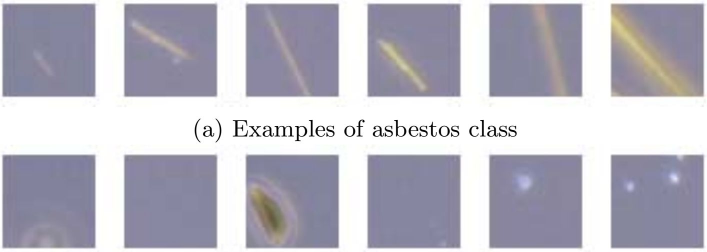
5.1 Image Dataset

In the experiments, we use 120 microscope images which contains 154 asbestos particles. To train the detector based on SVM, we pick up 21 images for training. The 600 positive examples are extracted from these 21 images. The 6000 negative examples without asbestos particles are also extracted from 21 images. The test set contains 137 asbestos particles in 99 images. Figure 4 shows the examples of training set. We understand the color with specific polarization and the needle-like shape of asbestos. On the other hand, the non-asbestos class includes various kinds of shapes and colors.

In this paper, the size of train images is set to 40×40 pixels. Thus, the size of detector is also 40×40 pixels. Our method determines that a local region of 40×40 pixels contains asbestos or not, and a Map image is constructed by the output values. If a test image contains asbestos particles of more than 40×40 pixels, the detector gives high output value when a part of large asbestos particle is included in a local region. Thus, the large asbestos particle appears in the Map image, and we can count the asbestos particles with various sizes accurately.

5.2 Evaluation Results

The evaluation result for the test set is presented in Table 1. The 1st row is the result of the proposed method based on the summation kernel of color and



(a) Examples of asbestos class

(b) Examples of non-asbestos class

Fig. 4. Examples of training images

Table 1. Evaluation result

	# Asbestos particles	# True positive (Rate)	# False positive
Proposed method	137	**122(89.1%)**	**15**
Joint feature	137	115(83.9%)	19
Only color feature	137	105(76.6%)	27

shape, and the 2nd row is for the detector using a kernel of joint feature of color and shape. The 3rd row shows the detector using only color histogram. # Asbestos particles shows the number of asbestos in test images. True positive means that an asbestos particle is classified correctly. The number in bracket shows the true positive rate. False positive means that a non-asbestos particle is mis-classified as the asbestos class. The detector with only color feature gives the worst result. By adding shape feature, the accuracy of counting is improved. However, the detector with a kernel of joint feature can not use the similarity of each feature effectively. Thus, the detector using the summation kernel shows the best performance. The accuracy of the proposed method achieves 89.1% while the number of false positives is the least. This shows the effectiveness of the detector with the summation kernel.

Figure 5 shows the examples of detection and counting result. The 1st row shows the detection results and the 2nd row shows corresponding Map images. Red rectangles are put to the asbestos particles detected by our detector. We understand that asbestos particles with various sizes are detected accurately.

Figure 6 shows a typical failure case by our approach. In Figure 6 (b), our method gives high values to two asbestos particles. However, since the two asbestos are too near, two particles are counted as one asbestos particle. In the detecting process, there is little case that our detector overlooks asbestos particles. Almost all of false negatives are classified into this kind of error. This shows the potential of our method. The improvement of counting process is a subject for future work.

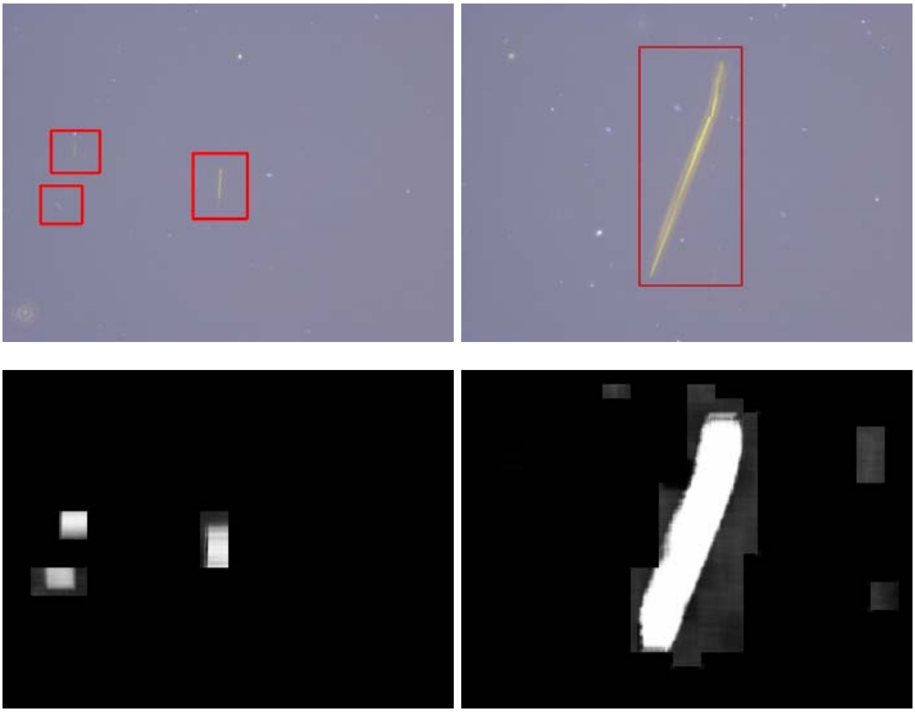
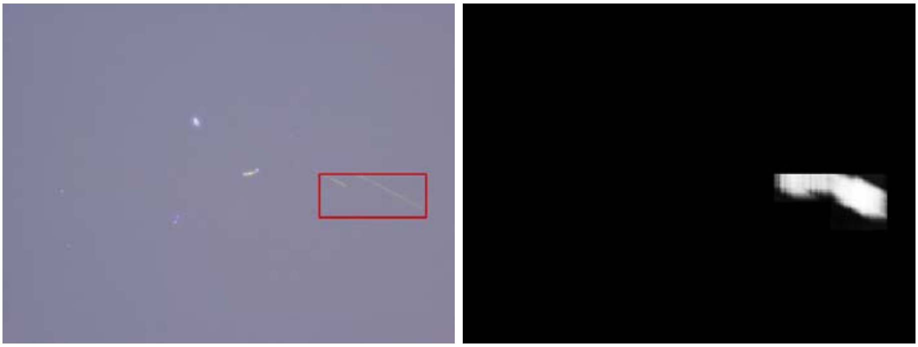


Fig. 5. Examples of detection and counting



(a) Detection result

(b) The corresponding Map image

Fig. 6. Typical failure case of our approach

6 Conclusion

In this paper, we have shown the automatic asbestos detection and counting method by computer. Since asbestos particles have a property in color and needle-like shape, we use color and shape feature. To use the similarity of each feature effectively, the kernels are applied to each type of feature independently, and the outputs of them are integrated by summation. The experimental results demonstrate that the proposed method gives the best result. It achieves about 90% with smallest number of false negatives. Since there are few researches about asbestos detection in building materials, this research will be very important for human health in the world.

Our method detects the asbestos particle with high probability. However, there is the case that failures are generated in counting process. The improvement of counting process is a subject for future work.

Acknowledgement

This research is supported by the research program (No.K1920 and No.K2061) of Ministry of the Environment of Japan.

References

1. Hotta, K.: Support Vector Machine with Local Summation Kernel for Robust Face Recognition. In: Proc. International Conference on Pattern Recognition, pp. 482–485. IEEE, Los Alamitos (2004)
2. Vapnik, V.: Statistical Learning Theory. John Wiley and Sons, Chichester (1998)
3. Haussler, D.: Convolution kernels on discrete structures, Technical report, UCSC-CRL-99-10 (1999)
4. Debnath, R., Takahashi, H.: Kernel Selection for the Support Vector Machine. IEICE Trans. Information and System E-87-D(12), 2903–2904 (2004)
5. Shawe-Taylor, J., Cristianini, N.: Kernel methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
6. Hjelmas, E., Low, B.K.: Face detection: A survey. Computer Vision and Image Understanding 83(2), 236–274 (2001)
7. Yang, M.-H., Kriegman, D., Ahuja, N.: Detecting faces in images: A survey. IEEE Trans. Pattern Analysis and Machine Intelligence 24(1), 34–58 (2002)
8. Papageorgiou, C., Poggio, T.: A Trainable System for Object Detection. International Journal of Computer Vision 38(1), 15–33 (2000)
9. Dalal, N., Trigs, B.: Histogram of oriented gradients for human detection. In: Proc. IEEE CS Conference on Computer Vision and Pattern Recognition, pp. 886–893 (2005)

Evaluation of Prediction Capability of Non-recursion Type 2nd-order Volterra Neuron Network for Electrocardiogram

Shunsuke Kobayakawa and Hirokazu Yokoi

Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology
2-4 Hibikino, Wakamatsu-ku, Kitakyushu-shi,
Fukuoka 808-0196, Japan
{kobayakawa-shunsuke@edu,yokoi@}life.kyutech.ac.jp

Abstract. The prediction accuracy of QRS wave that show electric excitement by the ventricle of the heart is low in linear predictions of electrocardiogram (ECG) used a conventional linear autoregressive model, and it is a problem that the prediction accuracy is not improved even if the prediction order is set second and third or more. The causes are that QRS wave generated by the nonlinear generation mechanism and the nonlinear components which the linear models cannot predict is included in ECG. Then, Non-recursion type 1st-order Volterra neuron network (N1VNN) and Non-recursion type 2nd-order Volterra neuron network (N2VNN) were evaluated about nonlinear prediction accuracies for ECG. The results of comparing nonlinear predictions of both networks showed that N2VNN is 17.6 % smaller about the minimum root mean square error indicating prediction accuracy than N1VNN.

1 Introduction

About the predictive coding of electrocardiogram (ECG), it is pointed out that the prediction error hardly decreases even if the prediction order is set second and third or more and that the compression performance is not improved. The cause is that it is not to be able to express the signal generation model by linear autoregressive models, for nonlinearity of ECG. Therefore, the nonlinear prediction based on nonlinear signal generation models is needed for the prediction of ECG.

For that reason, the prediction of ECG came to be done using Volterra functional series and neuron networks etc. which were nonlinear autoregressive models as the generation model for a nonlinear signal of ECG. However, these cannot finish decreasing sufficiently the prediction error in QRS wave which the nonlinearity is strong, for these are that capabilities of adaptation and learning are low in general.

It is known that the nonlinear time series signal is expressible by Volterra functional series consisting of addition of the calculated value using the time series signal value from past to present and the error which cannot be expressed by it. In the one that this is applied, there is a Volterra filter. Furthermore, there is a Volterra neuron network constructed of Volterra neurons with a built-in Volterra filter in an artificial neuron. The Volterra neuron network has been proposed as a neuron network which the processing performance of time series signals with strong nonlinearity is high by one of the authors,

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 679–686, 2009.

and the effectiveness has been confirmed by researches applied to each control of a humanoid robot [1], a myoelectric hand [2], skid of a car [3] and length system attitude of an aircraft [4] and elimination of artifacts in electroencephalogram etc.

Therefore, we propose a new neuron network for the nonlinear prediction of ECG in this paper. The prediction accuracy is evaluated by the experiment on nonlinear prediction for ECG using Non-recursion type 1st-order Volterra neuron network (N1VNN) and Non-recursion type 2nd-order Volterra neuron network (N2VNN).

2 Volterra Neurons

An artificial neuron with a built-in 1st-order Volterra filter is called non-recursion type 1st-order Volterra neuron (N1VN), and an artificial neuron with a built-in 2nd-order Volterra filter is called non-recursion type 2nd-order Volterra neuron (N2VN). These neurons use a similarity mapping as a judgment mapping as well as an artificial neuron.

2.1 I/O Characteristics of N1VN

Fig. 1 shows N1VN. The I/O characteristics of this neuron are shown in formulae from (1) to (3).

$$u^{(\tau)} = \sum_{i=1}^{n} w_i^{(\tau)} x_i^{(\tau)} \tag{1}$$

$$s^{(\tau)} = \sum_{p=0}^{Q} \sigma_p^{(\tau)} u^{(\tau-p)} - h^{(\tau)} \tag{2}$$

$$z^{(\tau)} = f(s^{(\tau)}) = A \tan^{-1}(s^{(\tau)}) \tag{3}$$

where x_i is the i^{th} input signal, w_i is the i^{th} connection weight, Q is the filter order, σ_p is the prediction coefficient corresponding to the signal obtained from between from the 1st delay element input to the Q^{th} delay element output. w_i, h and σ_p are changed by training.

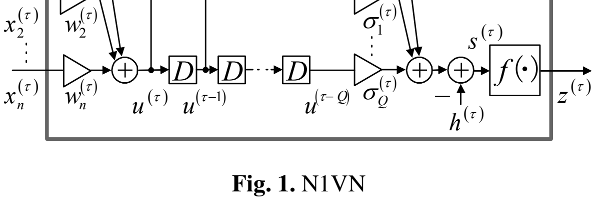


Fig. 1. N1VN

2.2 I/O Characteristics of N2VN

Fig. 2 shows N2VN. The I/O characteristics of this neuron are shown in the formulae from (**4**) to (**6**).

$$u^{(\tau)} = \sum_{i=1}^{n} w_i^{(\tau)} x_i^{(\tau)} \tag{4}$$

$$s^{(\tau)} = \sum_{p=0}^{Q} \sigma_1(p)^{(\tau)} u^{(\tau-p)} + \sum_{p=0}^{Q} \sum_{q=p}^{Q} \sigma_2(p,q)^{(\tau)} u^{(\tau-p)} u^{(\tau-q)} - h^{(\tau)} \tag{5}$$

$$z^{(\tau)} = f(s^{(\tau)}) = A \tan^{-1}(s^{(\tau)}) \tag{6}$$

where $\sigma_1(p)$ is the prediction coefficient of the 1st-order term corresponding to the signal obtained from between from the 1st delay element input to the Qth delay element output, $\sigma_2(p, q)$ is the prediction coefficient of the 2nd-order term corresponding to the product of all combinations of two signals included in combinations of the same signal obtained from between from the 1st delay element input to the Qth delay element output. w_i, h, σ_1 and σ_2 are changed by training.

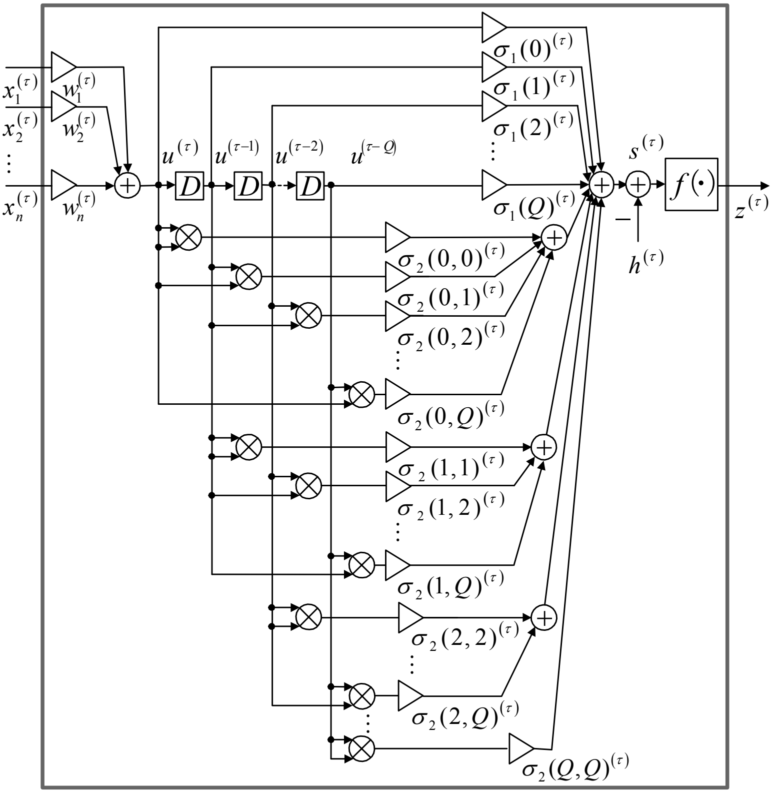


Fig. 2. N2VN

3 Nonlinear Predictions of ECG

The experiment for the evaluation of nonlinear prediction accuracies for ECG of two types which are N1VNN using N1VNs and N2VNN using N2VNs is performed. The experiment method and the results are shown as follows.

3.1 Experiment Method

1) The experiment for the evaluation of nonlinear prediction accuracies of networks of two types of N1VNN and N2VNN for ECG is performed. Fig. 3 shows a three-layer N1VNN or N2VNN of one input one output.
2) Each neuron network is trained using combinations of an input signal $x(\tau)$ in the time series pattern of one dimension in space direction and a teacher signal $y(\tau) = x(\tau+1)$. And the root mean square error (RMSE) calculated from the difference between the teacher signal and output signal of each neuron network is evaluated.
3) ECG signal of a healthy subject at rest is used for their training. Fig. 4 shows the ECG signal. This waveform is 415 data that are obtained from sampling frequency 100Hz. A pair of the input signal and the teacher signal is given once after 229 initial data are inputted into a neuron network at the training. This process is defined as one training cycle.
4) Table 1 shows the conditions of experiment for prediction accuracies evaluation of N1VNN and N2VNN. The initial values of the prediction coefficients use the exponential smoothing and the other initial values are decided by random numbers at the training process a time. The learning rule for each neuron network is the learning rule for the Volterra neuron network using gradient descent method.

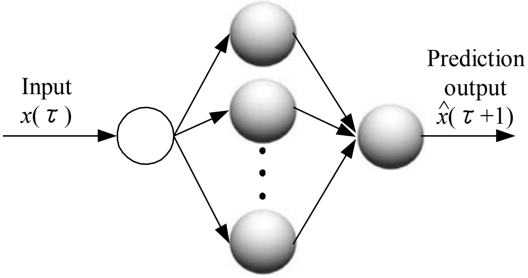


Fig. 3. Non-recursion type Volterra neuron network

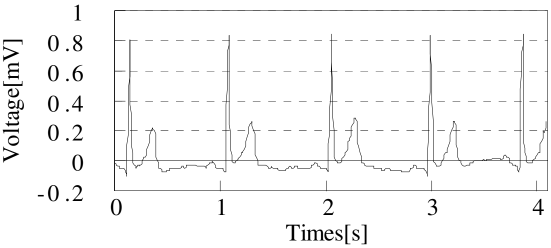


Fig. 4. ECG signal for the training

Table 1. Conditions of experiment for prediction accuracies evaluation of the neuron networks

Items of conditions / Types			N1VNN	N2VNN
Learning rules			Learning rule for the Volterra neuron network	
Initial conditions	Connection weights		-0.3 ~ 0.3	
	Thresholds		-0.3 ~ 0.3	
	Prediction coefficents	σ_p, σ_1	0.7× 0.3^p	
		σ_2	0.7× 0.3^p× 0.7× 0.3^q	
Middle layer elements		Range	2 ~ 50	
		Interval	2	
Filter length		Range	2 ~ 30	
		Interval	2	
Learning reinforcement coefficients	Gradient-based method	Range	10^{-6} ~ 1	
		Interval	10 times	
	Momentum		0	
Learning cycles			30,000	
Processing times			3	

5) The training is performed as an experiment condition setting parameters which are the middle layer elements, the filter length and the learning reinforcement coefficient. And averages of RMSEs obtained from search training of three times are compared.

3.2 Experiment Results

The Fig. 5 or 6 shows the minimum RMSEs in the search range of learning reinforcement coefficient and the standard deviations on the combination of the middle

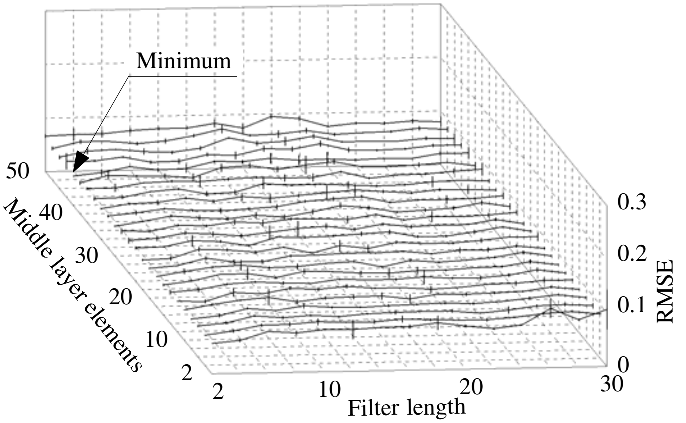


Fig. 5. RMSEs and the standard deviations of N1VNN

layer elements and the filter length of N1VNN or N2VNN obtained from the training.

The Fig.7 or 8 shows the output signal of N1VNN or N2VNN at the condition of the minimum RMSE obtained from the training. Table 2 shows the minimum RMSE and the condition which are obtained after the search training of N1VNN or N2VNN. This table shows that N2VNN is 17.6 % smaller about the minimum RMSE than N1VNN. As a result, it can be said that prediction capability of N2VNN is higher than one of N1VNN with the given ECG signal.

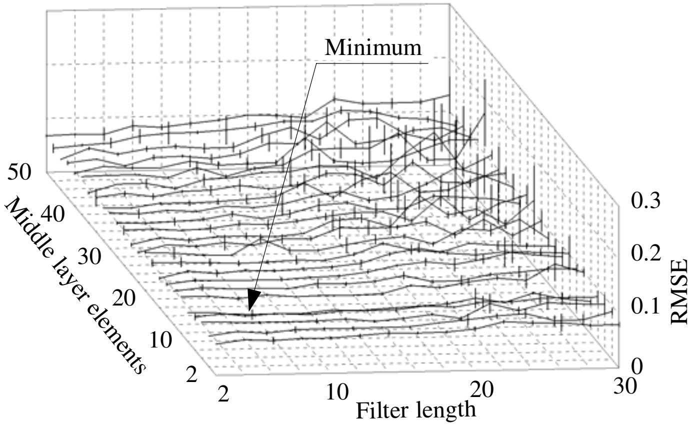


Fig. 6. RMSEs and the standard deviations of N2VNN

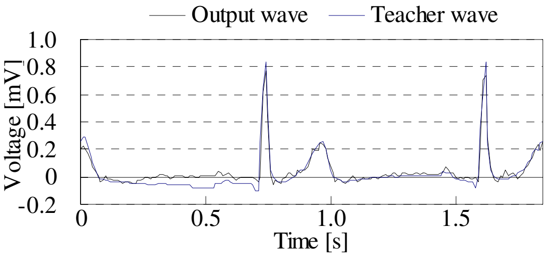


Fig. 7. The output signal of N1VNN at the condition of the minimum RMSE

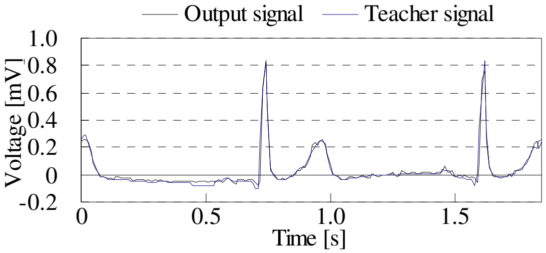


Fig. 8. The output signal of N2VNN at the condition of the minimum RMSE

Table 2. The minimum RMSEs and the conditions of N1VNN and N2VNN

Types / Items of data	N1VNN	N2VNN
Middle layer elements	42	10
Filter length	2	6
Learning reinforcement coefficient	1×10^{-2}	1×10^{-2}
RMSE	5.46×10^{-2}	4.50×10^{-2}
Standerd deviation	7.70×10^{-4}	2.30×10^{-3}
% of RMSE based on N1VNN	100	82.4

3.3 Discussion

The prediction accuracy when the minimum RMSE is obtained is considered. When a nonlinear output is expressed by the Volterra functional series, the expression error of this series becomes small as the degree of this series increase. Therefore, it is thought that the prediction accuracy of N2VNN improved because the expression capability for the nonlinear output of the built-in Volterra filter is higher.

The middle layer elements of N2VNN are a fewer than one of N1VNN. It also is thought that N1VNN covers the decrease in expression capability for the nonlinear output with increasing middle layer elements for the degree of the 1st-order Volterra filter is low though N2VNN can be few the middle layer elements because the 2nd-order Volterra filter is high performance.

The filter length of N2VNN is longer than one of N1VNN. This shows that the dependancy to the past data of the 2nd-order Volterra filter is higher. In addition, the expression error of a nonlinear output of the Volterra functional series becomes small as the filter length increases. However, the filter length of both NNs is short. The range of the effective past data to generate the prediction output is limited because the production mechanism in each part of ECG waveform is different respectively. Therefore, the prediction accuracy is down because the past data accumulated in the filter become noise components even if the filter length is lengthened, and the filter length shortened.

From above-mentioned, it is shown that the prediction capability of N2VNN is higher than one of N1VNN with the given ECG signal.

4 Conclusions

Result of comparing prediction capabilities of N1VNN and N2VNN using ECG of a healthy subject at rest, it is shown that the minimum RMSE of N2VNN is 17.6 % smaller and the prediction capability is higher than those of N1VNN.

Acknowledgments. We wish to express our gratitude to members in our laboratory who cooperate always in the academic activity.

References

1. Yoshikazu, F., Eiichi, I., Hirokazu, Y.: Robotic Control by Volterra Network. Technical Report of IEICE. NC 2003-78, 39–43 (2003)
2. Satoru, S., Hirokazu, Y.: A Motion Generating System for Multi-Fingered Myoelectric Hand: International Congress Series 1291. In: Kazuo, I., Kiyohisa, N., Akitoshi, H. (eds.), pp. 257–260. Elsevier, Amsterdam (2006)
3. Jun, M., Hirokazu, Y.: An Improvement of a Neural Network for Learning a Slip Angle of a Four-Wheel Steering Car. Technical Report of IEICE, NC2004-107, 87–90 (2004)
4. Shunsuke, K., Hirokazu, Y.: The Volterra Filter Built-in Neural Network for the Aircraft Pitch Attitude Control. In: The 58th Joint Conference of Electrical and Electronics Engineers in Kyushu, p. 429. JCEEE, Fukuoka (2005)

A New ART-LMS Neural Network for the Image Restoration

Tzu-Chao Lin, Mu-kun Liu, and Chien-Ting Yeh

Department of Computer Science and Information Engineering
Wufeng Institute of Technology, Chiayi, Taiwan 62153, R.O.C.
tclin@mail.wfc.edu.tw

Abstract. A novel neural network design–the adaptive resonance theory least mean square (ART-LMS) neural network–is proposed for the restoration of images corrupted by impulse noise. The network design is based on the concept of counterpropagation network (CPN). There is a vigilance parameter the ART network uses to automatically generate the cluster layer node for the Kohonen learning algorithm in CPN. In addition, the LMS learning algorithm is used to adjust the weight vectors between the cluster layer and the output layer for the Grossberg learning algorithm in CPN. The advantages of the ART-LMS network include an effective solution to the initial weight problem and a good ability to handle the cluster layer nodes for the CPN learning process. Experimental results have demonstrated that the proposed filter based on ART-LMS outperforms many well-accepted conventional as well as new filters in terms of noise suppression and detail preservation.

1 Introduction

Efficient removal of impulse noise from digital images has been important preprocessing tasks that must be carried out without harming the image details [1]. Neural networks have been a growing research interest in recent years, especially when it comes to nonlinear filtering techniques for image restoration. The network models include the backpropagation (BP), self-organizing feature map (SOFM) and counterpropagation network (CPN) models, etc. [2-6].

The competitive learning algorithm of the CPN Kohonen layer is a gradient-based unsupervised learning algorithm. It works best when the patterns are tightly clustered in distinct groups. However, under such a design, a neuron's initial weight vector can be located so far away from any input vectors that it is far from being competitive and therefore never learns. To make a difference, in this paper, we shall propose a new neural network called ART-LMS where the Kohonen layer and Grossberg layer in CPN are replaced by the adaptive resonance theory (ART) and least mean square (LMS) algorithm, respectively [7-8].

Since the center weighted median (CWM) filters fail to suppress noise to a satisfactory degree while preserving image details [10], we propose a neural-based CWM (NCWM) filter with an adjustable center weight. The adaptive weight for efficient removal of impulse noise without distorting image details that the proposed filter features is controlled by the ART-LMS neural network. The ART-LMS neural network is employed to obtain the optimal center weight. Experimental results demonstrate

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 687–694, 2009.

that the new filter based on the ART-LMS neural network outperforms many well-accepted conventional as well as new filters in terms of both noise suppression and detail preservation.

The rest of this paper is organized as follows. Section 2 is the illustration of the ART-LMS neural network. Section 3 describes how ART-LMS works to achieve image restoration. Then, Section 4 presents the results of some extensive experiments. Finally, Section 5 offers our conclusions.

2 ART-LMS Neural Network

Based on the framework of the CPN, we have developed the ART-LMS network to design the weight controller of the proposed NCWM filter. Figure 1 shows the topology of a three-layered ART-LMS network. The first layer is for input, the second is the competitive layer (ART layer), and the third is the layer of output nodes (LMS layer).The nodes in each layer are as Fig. 1 shows, fully interconnected to the nodes in the adjacent layer. The cluster layer can determine whether a new training pattern should be classified into a specific cluster or a new node should be automatically generated depending on the vigilance parameter. This way, unlike what happens with conventional unsupervised learning algorithms, the appropriate initial weights can be automatically obtained [11].

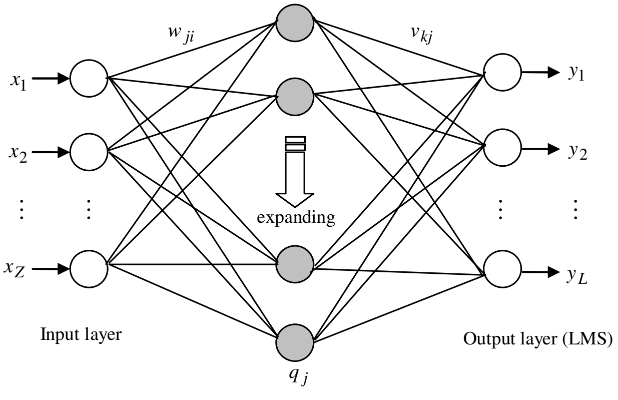


Fig. 1. the architecture of ART_LMS

The ART-LMS learning procedure includes two training processes. ART-LMS simultaneously trains the two weights through the three layers. The following is the ART-LMS algorithm.

Algorithm.

(1) Set the initial weights (first input vector and first target vector) and the learning rate.
Set vigilance δ.
Set the maximum epoch T, cluster node M and threshold θ.

```
(2) Do
    (a) Compute similarity $S_j$ of each cluster node
        $S_j = 1 - \left\| x - w_j \right\|_1$.
    (b) Decide the winner node $p$, where $p$ is the largest $S_j$.
    (c) If ( $S_p \geq \delta$ ) OR (number cluster nodes $> M$)
        then update weights.
        $w_{pi}(t+1) = \begin{cases} w_{pi}(t) + \alpha(n) \cdot (x - w_{pi}(t)), & w_{pi}(t+1) \geq 0 \\ 0, & w_{pi}(t+1) < 0 \end{cases}$
        $v_{kp}(t+1) = \begin{cases} v_{kp}(t) - \beta(n) \cdot | e(t) | \cdot | x - d |, & v_{kp}(t+1) \geq 0 \\ 0, & v_{kp}(t+1) < 0 \end{cases}$
    (d) If ( $S_p < \delta$ ) AND (number cluster nodes $< M$)
        then add one cluster node $J$.
        $w_{Ji} = x_i, (i = 1, \cdots, Z)$.
        $v_J =$ target vector $y_i, (i = 1, \cdots, L)$.
    (e) Increase epoch number $n$.
    Compute average difference of the distortion mean square error
        $AMSE = \frac{| MSE(n) - MSE(n-1) |}{MSE(n)}$.
While ( epoch number $n \leq T$ OR $AMSE > \theta$ )
```

In the ART-LMS learning algorithm, the first training vector x $(w_{ji} = x_i)$ and the target value v_{kj} are used directly to establish the first cluster node. Then, the next input training vector is compared with the first cluster node. It is assigned to the first cluster node if its similarity is larger than the vigilance parameter. Otherwise, a new cluster node is generated. That is, ART-LMS places the input vector into the most similar cluster node. This process is repeated for all training input vectors. Ultimately, ART-LMS can classify the X input patterns into M clusters (in general, $M < X$).

The nodes in the cluster layer compete (winner-take-all) for the input vector to be classified. The node with the largest similarity S_j is the winner and sends a signal 1 to the output layer. Then, only the weight vector w_{pi} from the winner (p-th node) in the cluster layer to the input layer is updated by using the following learning rule.

$$w_{pi}(t+1) = \begin{cases} w_{pi}(t) + \alpha(n) \cdot (x - w_{pi}(t)), & w_{pi}(t+1) \geq 0 \\ 0, & w_{pi}(t+1) < 0 \end{cases} \quad (1)$$

where the learning rate $\alpha(n)$ is a function of the learning epoch n, such as $\alpha(n) = \alpha_0 (1 - \frac{n}{T})$, defined with a predetermined constant $0 < \alpha_0 \leq 1.0$ and the total number of learning epochs T. Notably, the weight vectors to the loser nodes stay unchanged.

Meanwhile, only the weight in the output layer connected to the winner node is updated by using the least mean square (LMS) learning algorithm. The learning rule updates the weight v_{kp} from the output layer to the cluster node as follows.

$$v_{kp}(t+1)=\begin{cases} v_{kp}(t)-\beta(n)\cdot|e(t)|\cdot|x-d|, & v_{kp}(t+1)\geq 0 \\ 0, & v_{kp}(t+1)<0 \end{cases} \tag{2}$$

where the learning rate $\beta(n)$ is also a function of the learning epoch n, such as $\beta(n)=\beta_0(1-\frac{n}{T})$, defined with a predetermined constant $0<\beta_0\leq 1.0$ and the total number of learning epochs T. The variable x is the real corresponding input value, and the error $e(t)$ is the difference between the desired output d and the physical output y_k. In order to improve the convergent speed, the algorithm iterates the process until the average difference of the distortion mean square error ($AMSE$) falls below a threshold θ.

3 Image Restoration by ART-LMS

3.1 The Structure of the NCWM Filter

Let $C=\{(k_1,k_2)\,|\,1\leq k_1\leq H, 1\leq k_2\leq W\}$ denote the pixel coordinates of the noisy image corrupted by impulse noise, where H and W are the image height and width, respectively. Let $x(k)$ represent the input pixel value of the noisy image at location $k\in C$. At each location k, the observed filter window $w\{k\}$ whose size is $N=2n+1$ (n is a non-negative integer) is defined in terms of the coordinates symmetrically surrounding the input pixel $x(k)$.

$$w\{k\}=\{x_f(k): f=1,2,\cdots,n,n+1,\cdots,N\}, \tag{3}$$

where the input pixel $x(k)=x_{n+1}(k)$ is the center pixel. The output of the CWM filter is $y_c(k)=MED(w_c\{k\})$, where MED denotes the median operation and odd c $(c=1,3,5,\cdots,N)$ denotes the center weight. Now we have

$$w_c\{k\}=\{x_1(k),\cdots,x_n(k),c\Diamond x_{n+1}(k),x_{n+2}(k),\cdots,x_N(k)\}, \tag{4}$$

where $\Diamond$ represents the repetition operation. That is, the CWM filter outputs the median value of those $2n+c$ pixel values. However, the non-adaptive CWM filter processes the whole noisy image in a uniform way, making room for excessive or insufficient smoothing. To fix the drawback of lack of flexibility, the center weight c in $[0,N]$ is made adjustable within the filter window $w_c\{k\}$ by using the proposed ART-LMS network.

The framework of the proposed NCWM filter is illustrated in Fig. 2. It is composed of three parts: an observation vector for feature extraction, an ART-LMS weight controller, and a CWM filter. At first, according to the feature extraction result for the input vector, the ART-LMS weight controller gives the weight c to the CWM filter. To improve the filtering performance, the noise filtering procedure is progressively applied through several iterations.

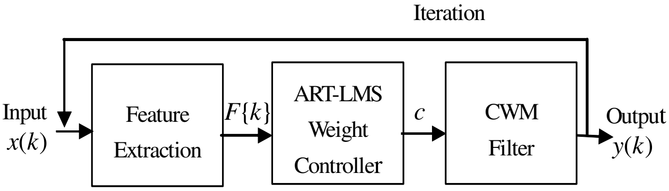


Fig. 2. The structure of NCWM filter

3.2 The ART-LMS Weight Controller

The following three variables can be defined to generate a feature vector $F\{k\}$ as the input data of the ART-LMS weight controller.

Definition 1: $a(k) = |x(k) - MED(w\{k\})|$. (5)

Definition 2: $b(k) = \dfrac{|x(k) - x_{c1}(k)| + |x(k) - x_{c2}(k)|}{2}$, (6)

where $|x(k) - x_{c1}(k)| \le |x(k) - x_{c2}(k)| \le |x(k) - x_i(k)|$, $1 \le i \le 2n+1$, i is not equal to $n+1, c1, c2$ [1, 12, 13].

Definition 3: $l(k) = |x(k) - w_3\{k\}|$. (7)

Notably, the values of $x_{c1}(k)$ and $x_{c2}(k)$ are selected to be the two closest pixel values to $x(k)$ in the filter window $w\{k\}$. In this paper, with the above three feature variables $a(k), b(k)$ and $l(k)$ playing their due roles, the feature vectors are given by

$$F\{k\} = \{a(k), b(k), l(k)\}. \quad (8)$$

To decide the adaptive weight c, we offer the ART-LMS weight controller that is put together on the basis of the ART-LMS network. It classifies the input feature vector and gives the weight v_j to its corresponding cluster according to the cluster layer and the output layer only one node in the ART-LMS network. Finally, the ART-LMS weight controller outputs the weight c.

The design of the optimal weight v_j, $j = 1, 2, \cdots, M$ for the NCWM filter concerns mainly with minimizing the mean square error (MSE). Here, M is the maximum cluster node in the training cluster layer. The learning rule updates the weights from the cluster layer to input layer using equation 1. The value of v_j can be trained independently by carrying out the LMS algorithm that is capable of minimizing the error function with respect to the cluster j [8]. The learning rule updates the weight v_j from the output layer to the cluster layer as follows.

$$v_j(t+1)=\begin{cases} v_j(t)-\beta(n)\cdot|\,e(t)\,|\cdot|\,x(k)-d)\,|, & v_j(t+1)\ge 0 \\ 0, & v_j(t+1)<0 \end{cases}. \tag{9}$$

The learning of ART-LMS can quickly converge toward the solution [9, 13].

4 Experimental Results

To demonstrate the effectiveness of the proposed NCWM filter based on ART-LMS network, some experiments have been conducted to check out the image restoration results. The random-valued impulse uniformly distributed over the range of [0, 255] is considered in 8-bit gray-scale images. Extensive experiments have been conducted on a variety of 512×512 test images to evaluate the performance of the proposed NCWM filter. The mean squared error (MSE) and the mean absolute error (MAE) have been employed to measure the restoration performance quantitatively. Smaller MSE and MAE values indicate better noise removal and image-detail preservation, respectively.

In the ART-LMS network training process, a training image 'Couple' corrupted by 20% impulse was used in the experiments. The network dynamically generated the cluster layer nodes in the training process. Figure 3 shows the relationship between MSE values and the number of nodes in the cluster layer of the ART-LMS network. The maximum number of clusters was set as 55 as Fig. 3 suggests. Figure 4 shows the training processes for different epoch numbers and their corresponding MSE values. The ART-LMS network converged when the number of training epochs reached six.

Table 1 compares the MSE and MAE results of removing impulse noise at 20%. As the table shows, the MSE and MAE values the NCWM filter gave are relatively much lower than those provided by the other filters for all test images. Notably, throughout all the experiments we collected the results after two iterations. Apparently, the NCWM filter is capable of producing better visual quality restored image by offering more noise suppression and detail preservation.

Table 1. Comparative restoration results in MSE and MAE for 20% impulse noise

Filters	Images									
	Goldhill		Boat		Lake		Couple		Lena	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
MED	69.51	4.84	62.89	3.92	106.69	5.71	99.11	5.43	43.69	3.46
CWM [10]	71.31	3.72	69.23	3.12	107.02	4.34	93.15	4.23	51.86	2.76
FM [12]	43.25	3.05	44.24	2.73	73.33	3.68	70.81	3.48	32.34	2.43
PFM [1]	36.76	2.52	36.05	2.24	62.81	3.16	64.94	2.86	24.66	1.94
ACWM [13]	36.08	2.41	35.31	2.05	61.48	2.89	61.43	3.02	25.26	1.81
CPN [4]	50.84	3.68	47.41	3.02	82.58	4.40	84.02	4.40	30.84	2.58
NCWM	35.14	2.35	34.12	2.00	58.74	2.79	59.92	2.98	23.42	1.74

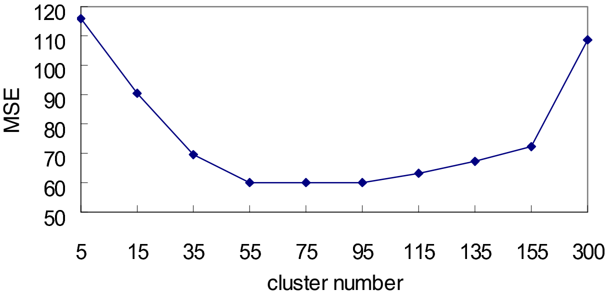


Fig. 3. MSE versus cluster number for the training process

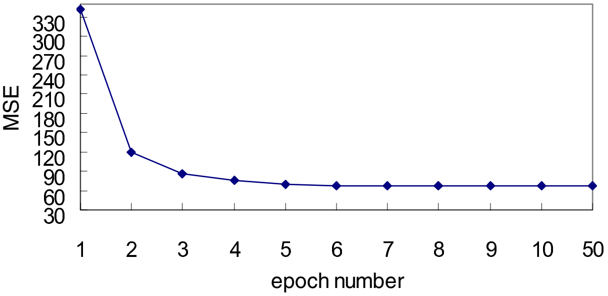


Fig. 4. MSE versus epoch number for the training process

5 Conclusions

In this paper, a novel neural network for image restoration is proposed, and a new filter built on top of the proposed ART-LMS network is presented to preserve more image details while effectively suppressing impulse noise. ART-LMS is a self-organizing neural network on the basis of adaptive resonance theory (ART) and least mean square (LMS) algorithm. ART-LMS uses the vigilance to dynamically create the cluster layer nodes. Thus, LMS algorithm is employed to obtain the optimal center weight for each cluster independently. Owing to the optimal weight of each cluster, the mean square error of the filter output can be minimized. The experimental results have demonstrated that the new NCWM filter is superior to a number of conventional as well as new filters in terms of the noise suppression and image detail preservation.

Acknowledgment. The author is grateful to the National Science Council of the Republic of China, for their support of this research under grant NSC-97-221-E274-010.

References

1. Lin, T.-C., Yu, P.-T.: Partition fuzzy median filter based on fuzzy rules for image restoration. Fuzzy Sets and Systems 147, 75–97 (2004)
2. Chun, S.H., Kim, S.H.: Data mining for financial prediction and trading: Application to single anf multiple markets. Expert Systems with Application 26, 131–139 (2004)

3. Kohonen, T.: Self-Organization Maps. Springer, Heidelberg (1995)
4. Hecht-Nielsen, R.: Application of counterpropagation networks. Neural Networks 1, 131–139 (1988)
5. Shi, S.M., Xu, L.D., Liu, B.: Improving the accuracy of nonlinear combined forcasting using neural networks. Expert Systems with Applications 16, 49–54 (1999)
6. Chang, F.J., Chen, Y.C.: A counterpropagation fuzzy-neural network modeling approach to real time streamflow prediction. Journal of Hydrology 245, 153–164 (2001)
7. Liu, T.-C., Li, R.-K.: A new ART-counterpropagation neural network for solving a forecasting problem. Expert Systems with Applications 28, 21–27 (2005)
8. Haykin, S.: Neural Networks A Comprehensive Foundation, 2nd edn. Prentice-Hall, Englewood Cliffs (1999)
9. Ohno, S., Sakai, H.: Convergence behavior of the LMS algorithm in subband adaptive filtering. Signal Processing 81, 1053–1059 (2001)
10. Ko, S.J., Lee, Y.H.: Center weighted median filters and their applications to image enhancement. IEEE Trans. Circuits and Systems. 38, 984–993 (1991)
11. Lin, T.-C., Yu, P.-T.: Centroid neural network adaptive resonance theory for vector quantization. Signal Processing 83, 649–654 (2003)
12. Arakawa, K.: Median filters based on fuzzy rules and its application to image restoration. Fuzzy Sets and Systems 77, 3–13 (1996)
13. Lin, T.-C.: A new adaptive center weighted median filter for suppressing impulsive noise in images. Information Sciences 177, 1073–1087 (2007)

Moving Vehicle Tracking Based on SIFT Active Particle Choosing

Tao Gao[1], Zheng-guang Liu[1], Wen-chun Gao[2], and Jun Zhang[1]

[1] School of Electrical Engineering and Automation, Tianjin University,
Tianjin, 300072, China
[2] Honeywell (China) Limited, Tianjin, 300042, China
gaotao231@yahoo.cn

Abstract. For particle filtering tracking method, particle choosing is random to some degree according to the dynamics equation, which may cause inaccurate tracking results. To compensate, an improved particle filtering tracking method is presented. A moving vehicle is detected by redundant discrete wavelet transforms method (RDWT), and then the key points are obtained by scale invariant feature transform. The matching key points in the follow-up frames obtained by SIFT method are used as the initial particles to improve the tracking performance. Experimental results show that more particles centralize in the region of motion area by the presented method than traditional particle filtering, and tracking results of moving vehicles are more accurate. The method has been adopted by Tianjin traffic bureau of China, and has a certain actual application prospect.

1 Introduction

Video surveillance system is an important part of Intelligent Transportation System (ITS). Detecting and tracking moving vehicles from video sequences is one of the important tasks in video surveillance system. Recently, many approaches have been proposed in this field. Ref.[1-2] use a template matching method to track the target. The blobs correspond to the moving target in the video sequences. But the method is difficult in handling scale change of the target, and threshold is subjectively determined with less robustness. Ref.[3] uses a snake model based tracking method which can reduce computational complexity and improve tracking accuracy. But it is sensitive to initialization and is difficult for actual application. Ref.[4-5] present a mean-shift method for motion tracking. Mean-shift method manifests high efficiency for target tracking with low complexity. But as a hill climbing algorithm, it may fall into a local minimum and lose the motion target when occlusion occurs. Ref.[6] uses Particle filtering to track moving targets; it is a successful numerical approximation technique for Bayesian sequential estimation with non-linear, non-Gaussian models. The basic Bayesian filtering is a recursive process in which each iteration consists of a prediction step and a filtering step. In this paper, the positions of particles are determined by key points obtained by scale invariant feature transform (SIFT) to improve the tracking efficiency, and we organize the paper as follows. A brief introduction on RDWT motion detection is given in Section 2. Scale invariant feature transform

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 695–702, 2009.

method is described in Section 3. The improved particle filtering tracking method is described in Section 4. Experimental results are reported in Section 5. Finally conclusions are summarized in Section 6.

2 Moving Vehicle Detection

In this paper, moving vehicles are detected by redundant discrete wavelet transforms (RDWT) [7], which conquer the drawback of time-domain methods. The RDWT is an approximation to the continuous wavelet transform that removes the down-sampling operation from the traditional critically sampled DWT to produce an over-complete representation. The shift-variance characteristic of the DWT arises from its use of down-sampling; while the RDWT is shift invariant since the spatial sampling rate is fixed across scale. As a result, the size of each sub-band in an RDWT is the exactly the same as that of the input signal.

Because the coefficients of the sub-bands of the redundant wavelet transform are highly correlated, and the direction and size are the same as the image, also, there is no translation in the sub-band; this paper uses the method which is based on redundant wavelet transforms to obtain the motion area. First, if the two adjacent frames are f_1 and f_2, we use the equation (1) to obtain the $MAS(x, y)$:

$$MAS\ (x, y) = \sum_{j=J_0}^{J_1} \begin{pmatrix} \left|LL_1^{(j)}(x, y) - LL_2^{(j)}(x, y)\right| + \left|LH_1^{(j)}(x, y) - LH_2^{(j)}(x, y)\right| \\ + \left|HL_1^{(j)}(x, y) - HL_2^{(j)}(x, y)\right| + \left|HH_1^{(j)}(x, y) - HH_2^{(j)}(x, y)\right| \end{pmatrix}. \tag{1}$$

The J_0 and J_1 are the starting and ending scales. We can obtain the motion area according to equation (2):

$$motion\ \ (x, y) = \begin{cases} 1 & MAS\ \ (x, y) \geq T\ \ \cdot \\ 0 & MAS\ \ (x, y) < T \end{cases} \tag{2}$$

T is the threshold which can be obtained automatically by otsu [8] method, then the mathematical morphology is used to remove noise points. The binary motion mask obtained from the redundant wavelet transforms can be considered as the original mask of the moving object. As the inner district of the object is usually flat and the characteristic is not obvious, this paper uses an assimilation method [9] to fill the mask. Figure 1 shows the motion vehicle detection result.

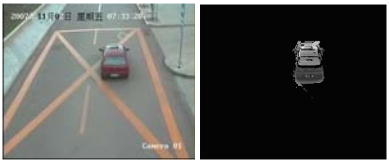


Fig. 1. Motion vehicle detection

3 Key Points of Vehicle Obtained

After getting the vehicle region, we use scale invariant feature transform (SIFT) to obtain the key points in the vehicle region. As SIFT transforms image data into scale-invariant coordinates relative to local features, an important aspect of this approach is that it generates large numbers of features that densely cover the image over the full range of scales and locations [10-11].

For image matching and recognition, SIFT features are first extracted from a set of reference images and stored in a database. A new image is matched by individually comparing each feature from the new image to this previous database and finding candidate matching features based on Euclidean distance of their feature vectors. There are four steps for this method.

1) Find Scale-Space Extrema. The only reasonable kernel for scale-space (continuous function of scale σ) is Gaussian:

$$G(x,y,\sigma)=\frac{1}{2\pi\sigma^2}e^{-(x^2+y^{2)})/2\sigma^2}. \tag{3}$$

For two-dimensional image, $L(x,y,\sigma)=G(x,y,\sigma)*I(x,y)$. Experimentally, Maxima of Laplacian-of-Gaussian: $\sigma^2\nabla^2 G$ gives best notion of scale. As LoG is expensive, we define Difference-of-Gaussians (DoG) instead:

$$\begin{aligned}&D(x,y,\sigma)=(G(x,y,k\sigma)-G(x,y,\sigma))*I(x,y)\\&=L(x,y,k\sigma)-L(x,y,\sigma)\end{aligned}. \tag{4}$$

The smoothed images need to be computed in any case for feature description, but in application we need only to subtract two images, and then choose all extrema within 3x3x3 neighborhood.

2) Remove the Instable and Edge Points. Take Taylor series expansion:

$$D(\vec{x})=D+\frac{\partial D^T}{\partial\vec{x}}\vec{x}+\frac{1}{2}\vec{x}^T\frac{\partial^2 D^T}{\partial\vec{x}^2}\vec{x}. \tag{5}$$

Minimize it to get true location of extrema:

$\hat{X}=-\frac{\partial^2 D^{-1}}{\partial X^2}\frac{\partial D}{\partial X}$, where $D(\hat{X})=D+\frac{1}{2}\frac{\partial D^{-1}}{DX}\hat{X}$.

We remove the instable points by the rule: $\left|D(\hat{X})\right|<0.03$, and also reject points which do not satisfy the equation (6):

$$\frac{(Tr(H))^2}{Det(H)}<\frac{(r+1)^2}{r}. \tag{6}$$

where $Tr(H)=D_{xx}+D_{yy}$, $Det(H)=D_{xx}D_{yy}-(D_{xy})^2$, r =10.

3) Orientation Assignment. We use scale of point to choose correct image:

$$L(x,y)=G(x,y,\sigma)*I(x,y). \tag{7}$$

Compute gradient magnitude and orientation using finite differences:

$$m(x,y)=\sqrt{\left(L(x+1,y)-L(x-1,y)\right)^2+\left(L(x,y+1)-L(x,y-1)\right)^2}. \tag{8}$$

$$\theta(x,y)=\tan^{-1}\left(\frac{\left(L(x,y+1)-L(x,y-1)\right)}{\left(L(x+1,y)-L(x-1,y)\right)}\right). \tag{9}$$

4) Obtain the Matching Points. The error of SIFT descriptors of the extrema points of image 1 and 2: $error=(\widehat{F}1-\widehat{F}2)^2$ can be used to obtain the matching points.

Figure 2 and 3 show the key points with orientation and SIFT matching result of a vehicle.

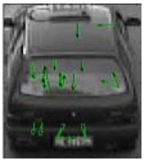


Fig. 2. Key points with orientation

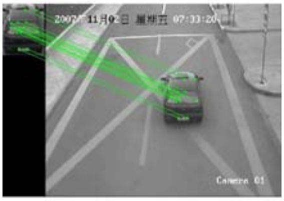


Fig. 3. SIFT matching result

4 Active Particle Filtering Combined with SIFT

Particle filtering [12-15] essentially combines the particles at a particular position into a single particle, giving that particle a weight to reflect the number of particles that were combined to form it. This eliminates the need to perform redundant computations without skewing the probability distribution. Particle filtering accomplishes this by sampling the system to create N particles, then comparing the samples with each other to generate an importance weight. After normalizing the weights, it resamples N particles from the system using these weights. This process greatly reduces the number of particles that must be sampled, making the system much less computationally intensive.

Particle Filtering estimates the state of the system, x and t, as time t as the Posterior distribution: $P(x_t \mid y_{0-t})$. Let $Es(t)=P(x_t \mid y_{0-t})$, $Es(1)$ can be initialized using prior knowledge Particle filtering assuming a Markov Model for system state estimation. Markov model states that past and future states are conditionally independent given current state. Thus, using Markov model, observations are dependent only on current state:

$$\begin{aligned} Es(t) &= P(x_t \mid y_{0-t}) = P(y_t \mid x_t, y_{0-t-1})P(x_t \mid y_{0-t-1}) = P(y_t \mid x_t)P(x_t \mid x_{t-1})P(x_{t-1} \mid y_{0-t-1}) \\ &= P(y_t \mid x_t)P(x_t \mid x_{t-1})Es(t-1) \end{aligned} \quad (10)$$

Final Result:

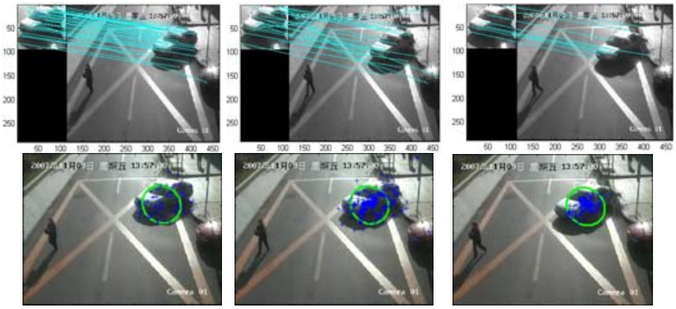
$$Es(t) = P(y_t \mid x_t)P(x_t \mid x_{t-1})Es(t-1) \quad (11)$$

Where: $P(y_t \mid x_t)$ is observation model and $P(x_t \mid x_{t-1})Es(t-1)$ is proposal distribution. The basic model usually consists of a Markov chain X and a possibly nonlinear observation Y with observational noise V independent of the signal X. System Dynamics Motion Model is $P(x_t \mid x_{0:t-1})$, and Observation Model is $P(y_t \mid x_t)$, Posterior Distribution is $P(x_t \mid y_{0\cdots t})$. Proposal Distribution is the Motion Model Weight, W_t = Posterior / Proposal = observation. Given N particles $\{x^{(i)}{}_{0:t-1}, z^{(i)}{}_{0:t-1}\}^{N}{}_{i=1}$ at time $t-1$, approximately distributed according to the distribution $P(dx^{(i)}_{0:t-1}, z^{(i)}_{0:t-1} \mid y_{1:t-1})$, particle filters enable us to compute N particles $\{x^{(i)}{}_{0:t}, z^{(i)}{}_{0:t}\}^{N}{}_{i=1}$ approximately distributed according to the posterior distribution $P(dx^{(i)}_{0:t}, z^{(i)}_{0:t} \mid y_{1:t})$. In video tracking, we do as follows: for each particle at time t, we sample from the transition beforehand. For each particle, we evaluate and normalize the importance weights, then multiply or discard particles with respect to high or low importance weights $W_t^{(i)}$ to obtain N particles. This selection step is what allows us to track moving objects efficiently. The state space is represented in the spatial domain as: $X = (x, y)$. We have initialized the state space for the first frame automatically by using the matching key points obtained by SIFT. A second-order auto-regressive dynamics is chosen on the parameters by SIFT matching to represent the state space (x, y). The dynamics is given as: $X_{t+1} = Ax_t + Bx_{t-1}$. Matrices A and B could be learned from a set of sequences where correct tracks have been obtained. In this paper, $A = \begin{bmatrix} 1 & \sigma \\ 0 & 1 \end{bmatrix}$, and $B = \alpha \begin{bmatrix} \frac{\sigma^3}{3} & \frac{\sigma^2}{2} \\ \frac{\sigma^2}{2} & \sigma \end{bmatrix}$, where $\sigma = 3$, and $\alpha = 0.35$. The observation y_t is proportional to the histogram distance between the color window of the predicted location in the frame and the reference color window: $Dist(q, q_X)$, Where q = reference color histogram, q_X = color histogram of predicted location. The following pseudo code depicts the overall structure of our tracking system.

Table 1. Pseudo code of tracking system

```
At time $t+1$, construct the $n^{th}$ of $N$ samples as follows:
    1. Generate a random number $r \in [0,1]$, uniformly distributed.
    2. Find, by binary subdivision on m, the smallest m for which $c_t^{(m)} \geq r$.
    3. First, draw a random variate $s_{t+1}^{(n)}$ from the density $p(x_{t+1} \mid x_t = s_t^{(m)})$,
       and then the first $k$ values of $s_{t+1}$ are set by matching key points of
       $SIFT(t, t+1)$.
    4. Store samples $n = 1, \cdots, N$ as $(s_{t+1}^{(n)}, \pi_{t+1}^{(n)}, c_{t+1}^{(n)})$ where
           $c_{t+1}^{(n)} = 0$
           $\pi_{t+1}^{(n)} = p(z_{t+1} \mid x_{t+1} = s_{t+1}^{(n)})$
           $c_{t+1}^{(n)} = c_{t+1}^{(n-1)} + \pi_{t+1}^{(n)}$
       and then normalize by dividing all cumulative probabilities $c_{t+1}^{(n)} = c_{t+1}^{(N)}$, that
    $c_{t+1}^{(N)} = 1$.
Mean properties can be estimated at any time $t$ as: $E[f(x) \mid z_t] \approx \sum_{n=1}^{N} \pi_t^{(n)} f(s_t^{(n)})$.
```

Figure 4 shows the results of particle filtering tracking which uses SIFT key points as initial particles.



Frame 1 Frame 5 Frame 8

Fig. 4. Particle filtering tracking based on SIFT matching

5 Experimental Results

In traditional particle filter tracking method, particle choosing is random according to the dynamics equation in some degree, which may cause inaccurate tracking results. In this section, we compare the tracking results between traditional particles filtering and our method showed in figure 5 and 6. The video is sampled at a resolution of 768x576 and a rate of 25 frames per second. The algorithms are tested on a 1400 MHz Celeron CPU, and software environment is VC++ 6.0. From the results we can see that when the scale of vehicle changes drastically, the particles still locate in the region of vehicle by our method; while for traditional particle filtering, particles obviously deviate from the vehicle which causes inaccurate results. The blue cross sign shows the particle, and red curve shows the motion track. We also sample 15 frames from the video sequence and compare the runtime of traditional particle filtering and our method (SIFT Particle Filtering), showed in figure 7; the difference is about 0.15s which can be ignored in actual application.

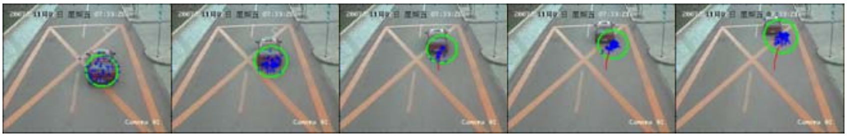


Fig. 5. Traditional particle filtering tracking (Frames 10, 24, 39, 56, and 60 are displayed)

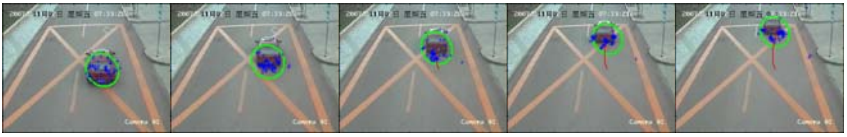


Fig. 6. Particle filtering tracking combined with SIFT by our method (Frames 10, 24, 39, 56, and 60 are displayed)

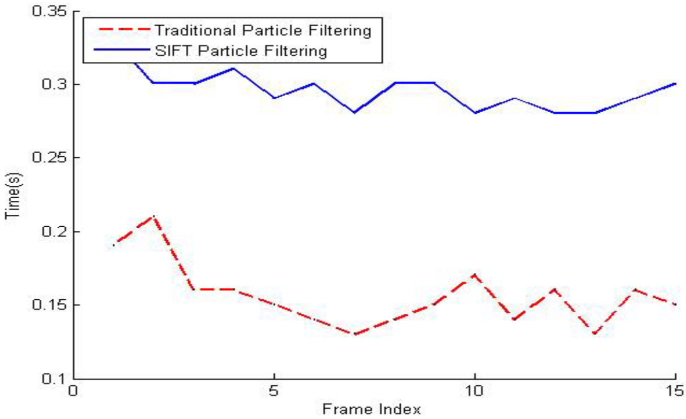


Fig. 7. Runtime comparison

6 Conclusions

In this paper, a novel moving vehicle tracking method based on particle filtering and SIFT is presented. First, the motion vehicle is detected by redundant discrete wavelet transforms, and scale invariant feature transform is used to extract the key points of vehicle; then according to SIFT matching, initial positions of particles can be obtained. By actively choosing the particles, tracking performance can be significantly improved. Our method has been adopted by traffic bureau, and has a certain actual application value.

Acknowledgments. The research is funded by Tianjin traffic bureau, China, and also funded by Tianjin Subway Safety System: [3240120], **Honeywell** (China) Limited.

References

1. Zhang, L., Han, J., He, W., Tang, R.S.: Matching Method Based on Self-adjusting Template Using in Tracking System. Journal of Chongqing University (Natural Science Edition) (06), 74–76 (2005)
2. Magee, D.R.: Tracking Multiple Vehicles Using Foreground, Background and Motion Models. Image and Vision Computing 22, 143–155 (2004)
3. Liu, H., Jiang, G., Li, W.: A Multiple Objects Tracking Algorithm Based on Snake Model. Computer Engineering and Applications 42(7), 76–79 (2006)
4. Comaniciu, D., Ramesh, V.: Mean Shift and Optimal Prediction for Efficient Object Tracking. IEEE Inter-national Conference on Image Processing 3, 70–73 (2000)
5. Comaniciu, D., Ramesh, V., Meer, P.: Kernel-based Object Tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(5), 564–577 (2003)
6. Hue, C., Le Cadre, J., Perez, P.: Tracking Multiple Objects with Particle Filtering. IEEE Transactions on Aerospace and Electronic Systems 38, 313–318 (2003)
7. Gao, T., Liu, Z.-g., Zhang, J.: BDWT based Moving Object Recognition and Mexico Wavelet Kernel Mean Shift Tracking. Journal of System Simulation 20(19), 5236–5239 (2008)
8. Otsu, N.: A Threshold Selection Method from Gray-Level Histogram. IEEE Trans.SMC. 9(1), 62–66 (1979)
9. Gao, T.: Liu, Z.-g.: Moving Video Object Segmentation based on Redundant Wavelet Transform. In: Proc. IEEE Int. Conf.on Information and Automation, pp. 156–160 (2008)
10. David, G.: Lowe: Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
11. David, G.: Lowe: Object Recognition from Local Scale-Invariant Features. In: International Conference on Computer Vision, Corfu, Greece, pp. 1150–1157 (1999)
12. Arulampalam, S., Maskell, S., Gordon, N., Clapp, T.: A Tutorial on Particle Filters for On-Line Nonlinear/ Nongaussian Bayesian Tracking. IEEE Trans. Signal Process. 50(2), 174–188 (2002)
13. Pitt, M., Shephard, N.: Auxiliary Particle Filters. J. Amer. Statist. Assoc. 94(446), 590–599 (1999)
14. Doucet, A., Vo, B.-N., Andrieu, C., Davy, M.: Particle Filter for Multi-Target Tracking and Sensor Management. In: The Fifth International Conference on Information Fusion, pp. 474–481 (2002)
15. Sidenbladh, H.: Multi-target Particle Filtering for the Probability Hypothesis Density. In: The Sixth International Conference on Information Fusion, pp. 1110–1117 (2003)

Classification of Fundus Images for Diagnosing Glaucoma by Self-Organizing Map and Learning Vector Quantization

Nobuo Matsuda[1], Jorma Laaksonen[2], Fumiaki Tajima[3], and Hideaki Sato[4]

[1] Oshima College of Maritime Technology, Information Science and Technology,
742-2193 Yamaguchi, Japan
matsuda@oshima-k.ac.jp
[2] Helsinki University of Technology, Laboratory of Computer and Information Science,
Fin-02015 Espoo, Finland
jorma.laaksonen@tkk.fi
[3] Yokohama National University, Education and Human Science,
240-8501 Yokohama, Japan
tajima@ynu.ac.jp
[4] Federation of National Public Service Personnel Mutual Aid Association
Kyousai Tachikawa HOSPITAL, 190-0022 Tokyo, Japan

Abstract. This paper presents a two stage diagnosis system that consists of Self-Organizing Map (SOM) and Learning Vector Quantization (LVQ) subsystems for diagnosis of fundus images. The first stage performs clustering and pseudo-classification of the input feature data by a SOM. The use of the pseudo-classes is able to improve the performance of the second stage consisting of a LVQ codebook. The proposed system has been tested on real medical treatment image data. In the experiments we have achieved a maximum accuracy rate of 71.2%, which is comparable to other results in literature.

1 Introduction

Fundus inspection is widely carried out to discover early stages of eye diseases such as glaucoma and retinal detachment. A survey report for 3,021 persons says that the frequency of such eye diseases reaches 5.5% in the male and 6.1% in the female [1]. The glaucoma results from increase of the intraocular pressure in excavation of the optic nerve papilla part and the papillae abnormality of the fundus. As a consequence it ends in blindness at the worst. For this reason, the glaucoma is a serious disease and becomes one of the important problems to be solved in the aging society to come. For the diagnosis of glaucoma, a general clinical method is the direct observation of the fundus or the observation of their photograph of the patient. Skills of a medical doctor are necessary for the decision of glaucoma, but it is still difficult for even an experienced doctor to diagnose accurately the condition of many examinees within a short time. Therefore, a quick and reliable diagnosis method is desired. Two samples of fundus images are shown in Figure 1. Up to now, attempts that try to detect from fundus images for the early stage of the glaucoma by image processing techniques have been made. We have earlier been developing a diagnosis system to detect

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 703–710, 2009.

excavatio papillae nervi optici from fundus images by an image processing technique [2]. Related studies on machine learning classifiers for diagnosing glaucoma have been made e.g. at University of California [3]. However, there is a problem that the detection often fails in the previous methods when the overlap between the classes of normal and abnormal data is large. This is often the case when there is no retina nerve bunch visible and the setting of a threshold is therefore difficult.

In this paper, we propose a technique that can steadily diagnose the glaucoma by the Learning Vector Quantization with aid of a clustering acquired on the Self-Organizing Map. The diagnosis is made by using feature vector data that can be simply gathered from the fundus images. The organization of this paper is as follows: Section 2 briefly describes the proposed diagnosis system. The Self-Organizing Map and the Learning Vector Quantization are described in Section 3. Experiment data, features and method parameters are described in Section 4. Experimental results are shown and the problems in the classification are discussed in Section 5. Finally, in Section 6 conclusions are drawn.

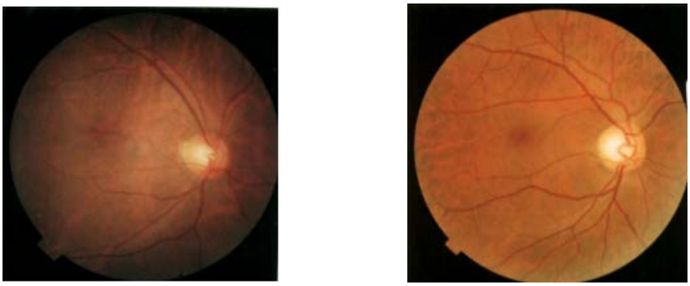


Fig. 1. Fundus images. Left is a normal fundus and right is an abnormal one.

2 Overview of Proposed Diagnosis System

The proposed diagnosis system has two stages. In the first stage, the Self-Organizing Map is used to cluster the feature vectors extracted from the training data. We then observe the clusters of the normal and abnormal samples on the SOM surface and define their division into *pseudo-classes*, currently two pseudo-classes for the normal and one for the abnormal fundus images. In the next stage, the medical treatment data is diagnosed with the help of that pseudo-class classification by using the Learning Vector Quantization method. The flow chart of the system is depicted in Figure 2.

The use of the pseudo-classes in classification is motivated as follows: The probability of abnormality rises if the size of the optici part is generally large. However the range of the extractable feature values of the normal samples is considerably larger than their mean values. This correlates with the complex distribution of the normal class on the SOM maps. We assume that the development of glaucoma is not a process where a member of a single normal group moves to one of a few abnormal groups. It is natural to recognize that it is rather a process where two or more kinds of normal and abnormal groups are involved.

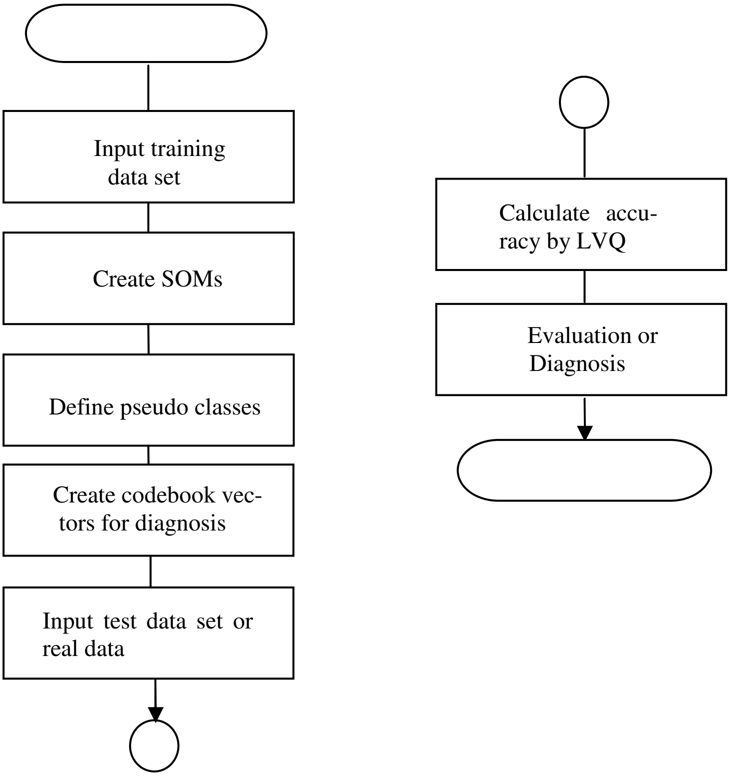

Input training data set
Create SOMs
Define pseudo classes
Create codebook vectors for diagnosis
Input test data set or real data
Calculate accuracy by LVQ
Evaluation or Diagnosis

Fig. 2. Flow chart of the proposed system

3 Self-Organizing Map and Learning Vector Quantization

The Self-Organizing Map (SOM) of Kohonen [5] is a nonlinear projection of a high-dimensional input data into a low-dimensional space. Many studies have been made on the clustering, visualization, and data mining capabilities of the SOM. These capabilities of the SOM attribute to acquiring useful information also in the medical area.

SOM is an unsupervised learning algorithm with a two-layer structure where all units in the self-organization layer are connected to all units in the input layer. Each input vector is during the training mapped to one of the output units placed in a two-dimensional grid. In the SOM training, the best-matching map unit, whose weight vector is nearest to the input vector, and its nearby units learn iteratively:

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)] \tag{1}$$

where $m_i(t)$ is the weight vector of the map unit i at time t, x(t) is the input vector with n-dimensions and $h_{ci}(t)$ is a neighborhood function, which includes the learning rate coefficient:

$$\begin{aligned} h_{ci}(t) &= \alpha(t) \qquad i \in N_c \\ h_{ci}(t) &= 0 \qquad\quad i \notin N_c \end{aligned} \tag{2}$$

N_c is the neighborhood of the best-matching map unit, and α(t) is the learning rate factor. As a result of the learning, the self-organizing layer forms a two-dimensional similarity relation which clusters the mutually similar feature vectors.

The Learning Vector Quantization (LVQ) [5] is a supervised version of vector quantization and aims at optimal classification of patterns in the input feature data. It moves the weight-vectors or prototypes to define optimal decision borders between the classes. In LVQ the prototype that most strongly reacts to the input data is trained. In the training, the class label of the trained prototype is compared with the class label of the input data vector. If the labels match, the prototype is moved towards the training sample, otherwise it is moved away from it. Both updates follow the formalism of Eq. (1), but now α(t) is negative for non-matching class labels and there is no neighborhood involved. The optimal classification borders are achieved by repeating the operation.

4 Experiment Data, Features and Method Parameters

4.1 Experimental Data, Feature Extraction and Selection

A series of experiments was conducted with fundus images produced by a clinical doctor. The total number of images is 133: 91 normal subjects and 42 abnormal ones. This data was divided into two separate subsets. The first subset for training consists of 46 normal persons and 21 abnormal ones and the other subset for classification test consist of 45 normal persons and 21 abnormal ones.

Colored fundus photographs of 24 bit RGB bitmaps were acquired with a scanner. The data used in our experiments are the R-, G- and B-pixel values of the fundus image on a horizontal line through the nipple center as shown in Figure 3. The scan lines consisted of 256 pixels. The white horizontal line in the center of the left image shows the scan line. These curves in the right graph show the values of the R-, G-, and B-pixels scanned on the horizontal line. The reason why we used such data is that it is easy to measure but bears relevance to a C/D ratio: The C/D ratio is often used in the conventional diagnosis of fundus images. It is the ratio of the diameter of the excavation cup and diameter of the optic nerve disk.

Different feature extraction methods were first applied to the 256-dimensional scan line pixel data. The best ones among these features were then selected by using leave one out (LOO) cross-validation in the training set and observing the equal error rate (EER) and the area under curve (AUC) values of the receiver operating curve (ROC). This step was performed by using the PicSOM system [7] in which parallel SOMs are used for scoring the similarity of image classes.

The B-pixel normalized value, G-pixel normalized value and the Fourier coefficient of the vectorial of the R- , G- and B-pixel were found to be the best features as the result of the cross validation test.

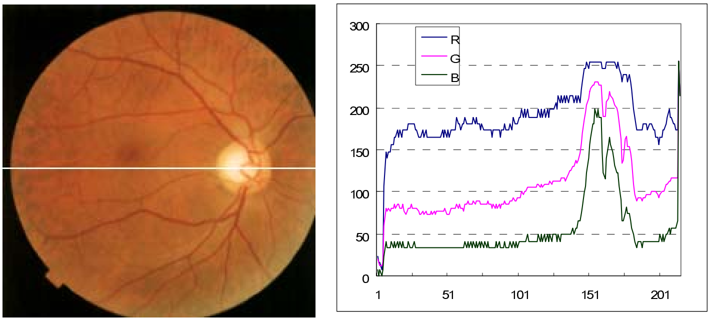


Fig. 3. Data sampling along the horizontal scan line

4.2 Parameters of SOM and LVQ

For the training of the SOM, we used the SOM Toolbox for Matlab 5 [8]. The parameters were automatically selected by the SOM Toolbox. For the classification, we used LVQ_PAK program package Version 3.1 [6], from which we used the programs LVQ2 and LVQ3. The main parameters used in our experiments are listed in Table 1.

Table 1. Parameters in LVQ

Number of codebook vectors	9
Running length in training	5,000
Initial learning rate	0.1
Window width in LVQ2	0.2
Number of neighbors in KNN classification	3

5 Results and Discussion

In this section, we will first show the results of data clustering with SOM maps and classification by LVQ. Then we will compare our system's performance with that of other relevant methods.

5.1 Clustering and Defining Pseudo-Classes with SOM Maps

Figure 4 shows a SOM map created with SOM Toolbox from normalized B-pixel values of the scan images.

The map can be interpreted as follows: (1) as a result of quantization, there exist units which consist of a single class and units consisting of overlapped classes. 14 units contain only the normal class, 8 units contain overlapped class, and 8 units contain only abnormal class. (2) There is an expanse area where only the normal class gathers in the region a little below the center of the map. Moreover, there is an area where the abnormal class gathers in the bottom left of the map. (3) On the other hand, some abnormal data are scattered in the upper part of the map which is overwhelmingly occupied by the normal class. These findings confirm our prior assumptions concerning the complex distribution of both the normal and the abnormal data, made in Section 2.

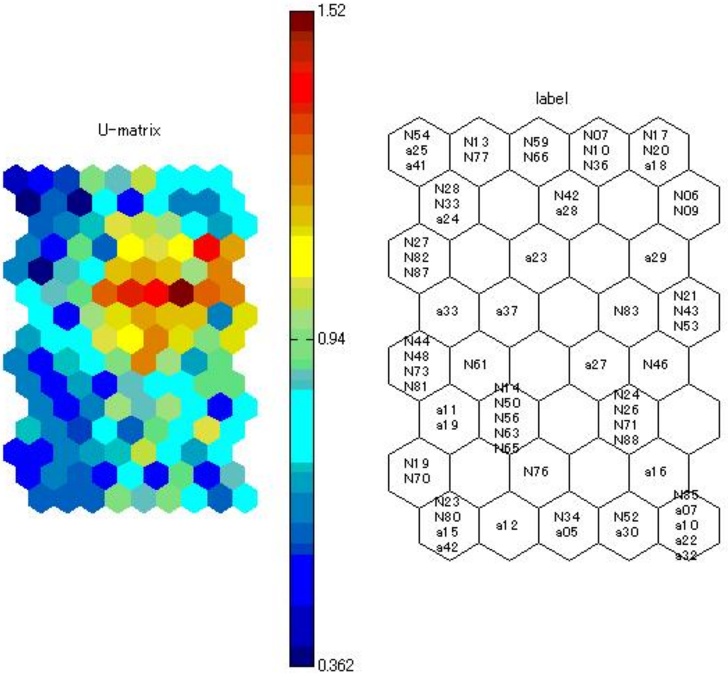


Fig. 4. SOM maps(size:6 x 8) distribution of normalized B-pixel data by SOM toolbox. The large 'N's indicate normal data samples, small 'a's abnormal ones.

5.2 Classification by LVQ

In our experiments, the maximum total accuracy was 71.21% in the proposed method case when two pseudo-classes were used for the normal class and one real class for the abnormal. In detail, the specificity (the accuracy for the normal class) was 97.78% and the sensitivity (the accuracy for the abnormal class) 14.29%. The total accuracy was also 69.7% in the case when the specificity was 91.11% and the sensitivity 23.81%. On the other hand, the total accuracy was 71.21% in the case when the pseudo-classes were not used. The specificity was 100.0% and then sensitivity was 9.52%. From these results, it seems evident that the use of the pseudo-classes contributes to the increased classification performance, especially for the relatively smaller class of abnormal data.

5.3 Selecting the Number of Pseudo-Classes

When the training data is divided into the pseudo-classes, we need a criterion for how to select the number of pseudo-classes separately for both the normal and abnormal real classes. The number of pseudo-classes should be larger than one for the pseudo-class technique to be effective. In our current experiments we have, due to the limited number of data available, used only two pseudo-classes for normal of the training classes. If more data were available, more pseudo-classes could be used. The division of pseudo-classes has been based on the result of one-dimensional SOM with two nodes. The size of each pseudo-class should be as equal as possible for improving the total accuracy.

5.4 Comparison with Other Methods for Diagnosis

The same or similar data has been used for evaluating the performance of different diagnosis systems. The accuracy values attained so far have been collected in Table 2. The value of SVM was obtained from the real medical treatment image data by support vector machines (SVM) (Kernel type of radial basis function was used). The specificity was 100.0%, but sensitivity was 0.0%.

The value of the method 1 was obtained by machine learning classifiers, such as Multilayer perceptrons, SVM, and mixtures of Gaussians. The results of two glaucoma experts were judged against the machine classifiers [3]. The value of the method 2 was obtained by SVM under the condition of 102 glaucoma fundus images and 54 normal ones [4]. It is difficult to directly compare the performance of our proposed diagnosis system with the accuracy of the conventional systems due to the differences in the data size, the features and analysis methods. The degradation mainly depends on not the adopted method itself, but on the data themselves. In particular, because of no screening of fundus images in our study, we used only a small quantity glaucoma data, and also other abnormalities such as ocular hypertension were contained in the abnormal class.

Table 2. The comparison with other methods

Method	Size of data sets	Total accuracy	Specificity	Sensitivity
Proposed method	**133**	**0.697**	**0.911**	**0.238**
SVM(RBF)	**133**	**0.682**	**1.00**	**0.0**
Two experts 1/2 [3]	345	0.843/0.749	0.75/0.88	0.96/0.59
Method 1 in [3]	345	0.839	0.790	0.900
Method 2 in [4]	156	0.855	0.764	0.904

6 Conclusions

In this paper, a two-stage diagnosis system that consists of SOM and LVQ was presented for diagnosis of fundus images. The first stage performs a pseudo-classification of feature values extracted from fundus images on the map. In the next stage, the fundus images were classified by LVQ as either normal or abnormal with aid of the information of the pseudo-classification. The performance of our system for diagnosis was verified by the actual diagnosis and treatment image data. Whenever the overlap between classes is large, the proposed system can increase classification performance, especially for the smaller class data.

References

1. Iwase, A., et al.: The Prevalence of Primary Open Angle Glaucoma in Japanese. The Tajimi Study Ophthalmology 111(9), 1641–1648 (2004)
2. Tajima, F., Sato, H., Miyatake, N., Matsuda, N.: Analysis of Fundus Images for Diagnosis of Excavatio Papillae Nervi Optici Focused on Optic Nerve Part its Depression Part. In: 18th Fuzzy System Symposium Proceedings, pp. 385–386 (2002)
3. Goldbaum, M.H.: Comparing Machine Learning Classifiers for Diagnosing Glaucoma from Standard Automated Perimetry. Investigative Ophthalmology & Visual Science 43(1), 162–169 (2002)
4. Hirahashi, H., et al.: Design of Fundus Image Analysis System for Glaucoma Diagnosis 3A1-3. In: The 20th Annual Conference of the Japanese Society for Artificial Intelligence (2006)
5. Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer, Tokyo (2001)
6. Kohonen, T., et al.: LVQ_PAK The Learning Vector Quantization Program Package Version 3.1 (1995)
7. Laaksonen, J., Koskela, M., Oja, E.: Class Distributions on SOM Surfaces for Feature Extraction and Object Retrieval. Neural Networks 17(8-9), 1121–1133 (2004)
8. Vesanto, J., et al.: SOM Toolbox for Matlab 5. Helsinki University of Technology, Finland (2000)

Facial Expression Recognition Techniques Using Constructive Feedforward Neural Networks and K-Means Algorithm

Liying Ma

Dept. of Applied Computer Science, Tokyo Polytechnic University
Atsugi, Kanagawa, 243-0297, Japan

Abstract. In this paper two facial expression recognition (FER) techniques are proposed. Lower-frequency 2-D DCT coefficients of binarized edge images are utilized in both methods as features for recognition. The first approach uses a constructive one-hidden-layer (OHL) feedforward neural network (OHL-NN) and the second approach is based on the K-means algorithm as classifiers. The 2-D DCT is used to compress the binarized edge images to capture the important features for recognition. Facial expression "neutral" is regarded as a subject of recognition in addition to two other expressions, "smile" and "surprise". The two proposed recognition techniques are applied to two databases which contain 2-D front face images of 60 men (database (a)) and 60 women (database (b)), respectively. Experimental results reveal that the proposed two techniques yield performances that are comparable to or better than that of two other recognition methods using vector matching and fixed-size BP-based NNs, respectively. The first proposed method yields testing recognition rates as high as 100% and 95%, and the second one achieves as high as 100% and 98.33%, for databases (a) and (b), respectively.

1 Introduction

Facial expressions play an important role in human communications, since they carry much information about humans, such as one's feelings, emotions and so on. Computer-based automatic facial expression recognition (FER) can help to create human-like robots and machines that are expected to enjoy truely intelligent and transparent communications with humans. To date, many FER methods have been proposed in the literature (see for example, [1]-[10] and the references therein). A good review can be found in [4] and [5].

Ekman developed a facial action coding system (FACS) [1] for facial expression description, which is a pioneer system that has brought great influence on later research and development in the field. The FACS deals with 3-dimensional (3-D) facial models where the entire face is divided into 44 action units (AUs), such as nose, mouth, eyes, etc.. The muscles movements of these feature-bearing AUs are used to describe any human facial expression of interest. A drawback of the FACS is that it requires 3-dimensional measurements and is thus too complex for real-time processing. A modified FACS processing only 17 important AUs

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 711–719, 2009.

is proposed in [2] for facial expression analysis and synthesis. However, 3-D measurements are still needed.

Recently, FER using 2-D digital images has been a focus of research [3]-[10]. In [3], a radial basis function (RBF) neural network (NN) is proposed for FER based on motion. The 2-D discrete cosine transform (2-D DCT) is used to compress an entire difference face image between a “neutral” and an expression image, and the resulting lower-frequency 2-D DCT coefficients are used to train a one-hidden-layer NN (OHL-NN) using BP-based training algorithm or constructive algorithm in [6,7]. In the above FER methods, the facial expression, “neutral”, is usually not regarded as a subject of recognition. Instead it is used as a reference for the detection of motion of other expressions such as “smile”, “surprise”, etc. As reference the neutral images can significantly facilitate the recognition of other expressions. However, “neutral” should be a subject of recognition as it represents the same importance and amount of information as other expressions do in human communications. When “neutral” is considered as a subject of recognition task, the above-mentioned techniques are no longer applicable, or present very poor performance. Refer to [10] where the recognition of “neutral” is considered and a vector matching based technique was proposed.

More recently, an interesting FER technique is proposed in [5] that uses the line-based caricatures. In this method, the normally binarized edges of facial images are connected and thinned to generate a line edge map (LEM) which is compared with the line-based caricatures pre-sketched manually to perform the recognition task. The expression “neutral”, “smile” and “surprise” are considered as subjects of recognition. The technique is computationally efficient but the performance needs to be improved, as the averaged recognition rate is only 86.6% for the AR face database [5] which is almost the same in size as the databases that are used in our work.

The recognition methods using fixed-size NNs have been found to be particularly promising [3,6,4], However, determining a proper network size has always been a frustrating and time consuming experience for NN developers. Constructive NNs have been used to overcome this difficulty [11,12,13]. In the constructive training of a OHL-NN, a small network with a few hidden units, say one or two, is first trained by a BP-type algorithm, and then new hidden units are added to the existing network one at a time until the network performance is maximized. See [11,12,13] for other constructive NNs concepts and architectures.

In normal vector matching, only a single standard vector is extracted for each expression category. According to our experience this usually can not handle the varieties of an expression [8]. This may become more significant when the neutral images are regarded as subjects of recognition rather than reference images for forming difference images [7]. Using multiple standard vectors for an expression may mitigate this problem. The K-means algorithm is a powerful tool for identifying multiple standard vectors.

In this paper, we propose two new FER techniques that use the binary edge images, 2-D DCT, a constructive OHL-NN and the K-means algorithm as classifiers, respectively. The two proposed recognition techniques are applied to two databases which contain 2-D front face images of 60 men (database (a)) and 60 women (database (b)), respectively. Experimental results reveal that the proposed techniques are comparable to or better than two other recognition methods that use normal vector matching and fixed-size BP-based NNs, and the K-means based classifier outperforms the OHL-NN based technique. The former method yields testing recognition rates as high as 100% and 95%, and the latter technique achieves recognition rates as high as 100% and 98.33%, for databases (a) and (b), respectively.

2 Two Novel Facial Expression Recognition Techniques

New recognition techniques that use binary edge images, 2-D DCT and constructive OHL-NNs and K-means are depicted in Fig. 1. Feature extraction at each preprocessing step and classification process are described in detail below.

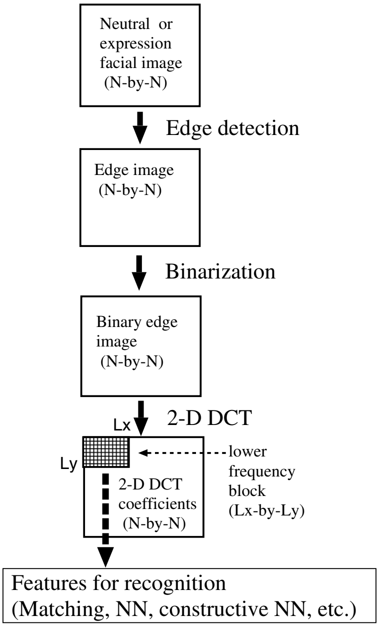

Neutral or expression facial image (N-by-N)
Edge detection
Edge image (N-by-N)
Binarization
Binary edge image (N-by-N)
Lx
2-D DCT
Ly
2-D DCT coefficients (N-by-N)
lower frequency block (Lx-by-Ly)
Features for recognition (Matching, NN, constructive NN, etc.)

Fig. 1. Feature extraction flow of the proposed technique

(A) Edge detection

Facial expressions are embodied by the AUs movements which are mainly characterized by the movements of edges within the images. These edges are expected to contain much information useful for the identification of facial expressions. Detection of edges from facial images is the first step towards the recognition stage. In general, any edge detector may serve this purpose. In this work we utilized the Sobel filter. The horizontal and vertical edges at pixel (i, j) are denoted by $g_h(i, j)$ and $g_v(i, j)$, respectively. Our experiments have shown that the following edge definition tends to produce the best recognition results

$$g(i, j) = max(|g_h(i, j)|, |g_v(i, j)|) \tag{1}$$

(B) Binarization of edge images

The detected edges could be used as is for our recognition purpose, but extensive experiments have revealed that this is not a good idea since the recognition rates obtained by vector matching are extremely poor. The detected edges are therefore binarized by using a proper threshold, which is calculated over those non-zero pixels of the edge image.

(C) Application of 2-D DCT and features for recognition

The binarized edges detected as features for our recognition purpose are represented by a large number of binary data (N-by-N) where many of these data may provide no contribution to the recognition performance and should therefore be filtered out and screened in a proper way. Consequently, we propose to apply the 2-D DCT to compress the binary edge images and use the lower-frequency DCT coefficients as the processed final features for the classification that follows. A lower-frequency block of size $L_x \times L_y$ is utilized. A vector arranged from this block is normalized and arranged as a feature vector for classification. The block size that leads to the best recognition rate may be searched and chosen experimentally.

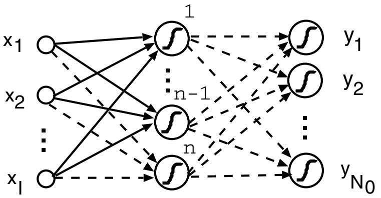


Fig. 2. Structure of a constructive OHL-NN with nonlinear output nodes (solid lines indicate the trained and then fixed weights, dotted lines indicate the weights to be trained in the input-side and output-side training phases)

(D) An FER technique based on constructive OHL-NN

A block diagram for our proposed constructive OHL-NN is shown in Fig. 2. The input vector to the network is denoted by $\mathbf{x}^j = (x_1^j, x_2^j, \cdots, x_I^j)^T$, and the output target of the network is denoted by $\mathbf{d}^j = (d_1^j, d_2^j, \cdots, d_{N_0}^j)^T$, $j = 1, 2, \cdots, P$, where I $(= L_x \times L_y)$ and N_0 are dimensions of input and output vectors of the network, respectively, and P is the number of training samples. In this paper, the sigmoidal function $f(\cdot)$ is used as the activation function of both the hidden and the output units of the OHL-NN. The input-side training of the n-th hidden unit is accomplished based on maximizing the following correlation-type performance index [12]

$$J_{input} = \sum_{o=1}^{N_0} \left| \sum_{j=1}^{P} (e_{n-1,o}^{j} - \bar{e}_{n-1,o})(f(s_n^j) - \bar{f}) \right| \tag{2}$$

where $e_{n-1,o}^{j} = y_o^j(n-1) - d_o^j$, $\bar{e}_{n-1,o} = \frac{1}{P}\sum_{j=1}^{P} e_{n-1,o}^{j}$, and $\bar{f} = \frac{1}{P}\sum_{j=1}^{P} f(s_n^j)$. The "quickprop" algorithm [12] is used to obtain the maximization of the above objective function. The output-side training is performed by minimizing a summed squared error criterion. A second-order algorithm, such as the Quasi-Newton algorithm [11,13] may be used to solve this nonlinear optimization problem. Following the above two training phases, the testing data that is unseen in the training phase is fed to the trained network to evaluate its generalization capabilities. "Winner-take-all" criterion is applied to the output nodes to obtain classification. That is, a node that achieves the largest value is regarded as the winner, and the category of the input image being considered will be classified as the expression the winner node corresponds to.

(E) An FER technique based on K-means algorithm

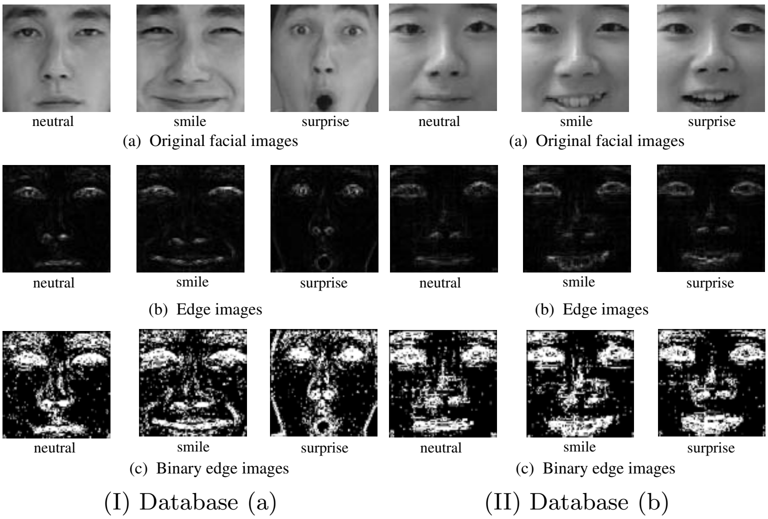
Classification may be simply performed based on vector matching, with the squared errors being used as criterion. A standard vector is obtained in training, which is a global centroid of training vectors of the same expression category.

Our extensive experimental simulations have revealed that the expressions of interest may have many distinct subgroups (clusters), and their recognition may be improved if one categorizes these subgroups or their corresponding standard vectors into the classification process. To this end, one needs to use a tool to specify or search for the members of each subgroup. In this work, the K-means algorithm is used for this purpose. In each search, the initial vectors (centroids) for each subgroup are randomly selected from the training vectors, and the nearest-neighbor (minimum distance) rule is utilized in classifying the group members. For a user-specified number of subgroups, a number of independent search runs are performed, and the outcome of a run with minimum summed distance is taken as the search result. The number of independent search runs is set sufficiently large such that the outcome remains unchanged. Because the dimension of training vectors is small, the K-means algorithm converges very fast. The centroids for all the subgroups are regarded as the standard vectors for recognition. It should be noted that in this work the global centriod for each expression is used as the first standard vector without exception.

3 Experimental Results

In this work, four (4) recognition techniques are applied to two databases, namely database (a) and (b) that have front face images of 60 men and 60 women, respectively. The first is a normal vector matching classifier (Method A), the second is implemented by fixed-size BP-based NNs (Method B), the third is our first new technique based on the constructive OHL-NNs (Method C), and the fourth is the second new method based on K-means algorithm (Method D).

Three facial expression images ("neutral", "smile", and "surprise") of size 128 × 128 for each individual are the subjects of recognition. Sample images from databases (a) and (b) are given in Fig. 3, where their edge and binarized edge images are also provided. For each database the images of the first 40 individuals are used for training and the remaining images for 20 individuals are used to test the trained recognition system. Some typical results are shown below.



(a) Original facial images (a) Original facial images

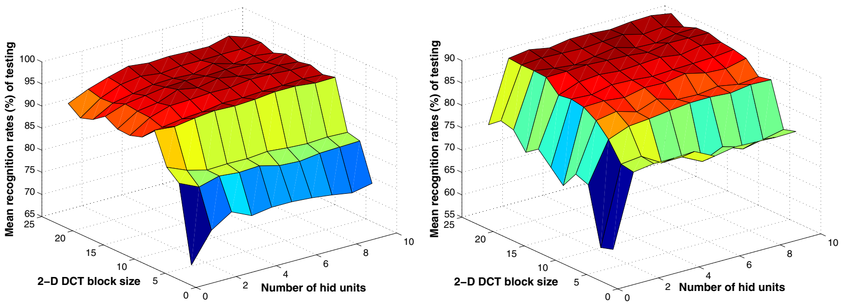
(b) Edge images (b) Edge images

(c) Binary edge images (c) Binary edge images

(I) Database (a) (II) Database (b)

Fig. 3. Samples from databases (a) and (b) and their edge and binarized edge images

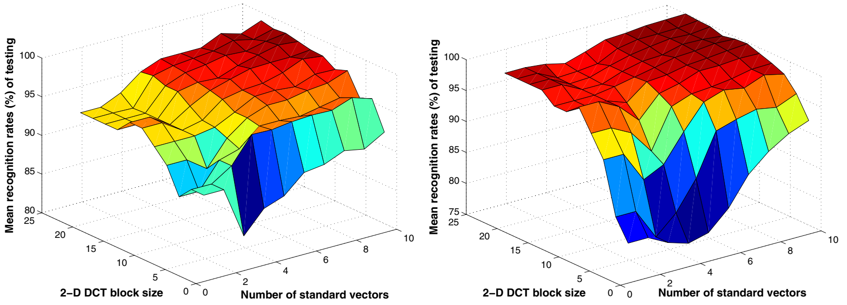
The mean testing recognition rates produced by Methods C for databases (a) and (b) are provided in Fig. 4 versus the the block size and the number of hidden units. Fig. 5 presents similar results by Method D for databases (a) and (b) versus the block size and the number of standard vectors. Comparisons among the four methods are indicated in Fig. 6, where the maximum rates of testing are shown (for Method A, maximum rates are the same as the mean ones). In Method C, for each block size and each number of hidden unit(s), 20 networks with different random initial weights were generated. Similarly, in Method D, for each block size and each number of standard vector(s), the K-means algorithm was run 20 times from different random initial centroids. The mean and maximum recognition rates for training and testing are extracted from these networks and runs. From Figs. 4-6, one sees that the proposed techniques (Methods C & D) provide results that are better than or similar to those produced by



(I) Database (a)

(II) Database (b)

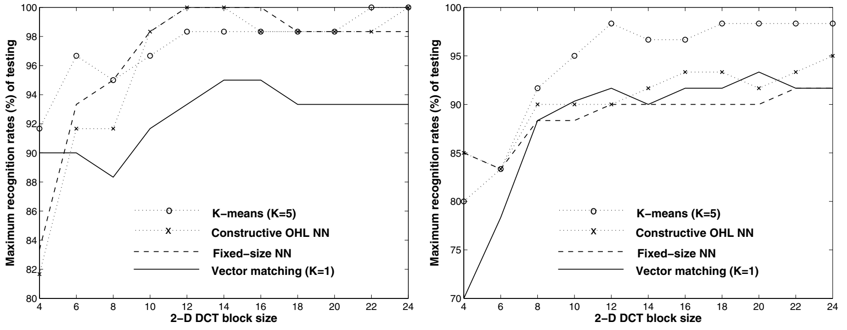
Fig. 4. Mean testing recognition rates of Method C versus the block size and the number of hidden unit(s)



(I) Database (a)

(II) Database (b)

Fig. 5. Mean testing recognition rates of Method D versus the block size and the number of standard vector(s)



(I) Database (a)

(II) Database (b)

Fig. 6. Maximum testing recognition rates of Methods A-D versus the block size (K=1: a single standard vector; K=5: five standard vectors, for each expression)

Methods A & B. It should be noted that the performance of Methods B, C, and D tends to be influenced by the training parameter settings and data arrangements. Further detailed experimental trials are needed to fully evaluate these results.

4 Conclusions

In this paper, two novel facial expression recognition methods are proposed, where "neutral" is treated as a subject of recognition in addition to two other expressions "smile" and "surprise". The proposed techniques have been compared to vector matching and the fixed-size BP-based NNs. These four (4) recognition methods have been applied to two databases, and it has been discovered that the proposed techniques present some performance advantage over other two methods. Future topics include 1) performing further detailed experimental trials, 2) investigating the capabilities of the proposed methods in case of larger number of facial expressions, and 3) applying the proposed techniques to other images such as the AR face database etc. to conduct a comparative study.

References

1. Ekman, P., Friesen, W.: Facial Action Coding System. Consulting Psychologists Press (1977)
2. Kawakami, F., Yamada, H., Morishima, S., Harashima, H.: Construction and Psychological Evaluation of 3-D Emotion Space. Biomedical Fuzzy and Human Sciences 1(1), 33–42 (1995)
3. Rosenblum, M., Yacoob, Y., Davis, L.S.: Human expression recognition from motion using a radial basis function network architecture. IEEE Trans. on Neural Networks 7(5), 1121–1138 (1996)
4. Pantic, M., Rothkrantz, L.J.M.: Automatic analysis of facial expressions: the state of the art. IEEE Trans. Pattern Analysis & Machine Intelligence 22(12), 1424–1450 (2000)
5. Gao, Y.S., Leung, M.K.H., Hui, S.C., Tananda, M.W.: Facial expression recognition from line-based caricature. IEEE Trans. System, Man, & Cybernetics (Part A) 33(3), 407–412 (2003)
6. Xiao, Y., Chandrasiri, N.P., Tadokoro, Y., Oda, M.: Recognition of facial expressions using 2-D DCT and neural network. Electronics and Communications in Japan, Part 3 82(7), 1–11 (1999)
7. Ma, L., Khorasani, K.: Facial expression recognition using constructive feedforward neural networks. IEEE Trans. System, Man, and Cybernetics (Part B) 34(4), 1588–1595 (2003)
8. Ma, L., Xiao, Y., Khorasani, K., Ward, R.: A new facial expression recognition technique using 2-D DCT and K-means algorithms. IEEE Intl. Conf. on Image Processing 2, 1269–1272 (2004)

9. Xiao, Y., Ma, L., Khorasani, K.: A new facial expression recognition technique using 2-D DCT and neural network based decision tree. In: Intl. Joint Conf. on Neural Networks, pp. 4728–4735 (2006)
10. Inada, Y., Xiao, Y., Oda, M.: Facial expression recognition using vector matching of spacial frequency components. IEICE Technical Meeting, DSP2001-103, pp. 25–32 (2001)
11. Kwok, T.Y., Yeung, D.Y.: Objective functions for training new hidden units in constructive neural networks. IEEE Trans. on Neural Networks 8(5), 1131–1148 (1997)
12. Fahlman, S.E., Lebiere, C.: The cascade-correlation learning architecture, Tech. Rep. CMU-CS-90-100, Carnegie Mellon University (1991)
13. Ma, L., Khorasani, K.: New training strategies for constructive neural networks with application to regression problems. Neural Networks 17, 589–609 (2004)

A Neural Network Based Classification of Human Blood Cells in a Multiphysic Framework

Matteo Cacciola, Maurizio Fiasché, Giuseppe Megali,
Francesco C. Morabito, and Mario Versaci

University Mediterranea of Reggio Calabria, DIMET,
Via Graziella Feo di Vito, I-89100 Reggio Calabria, Italy
{matteo.cacciola,maurizio.fiasche,giuseppe.megali,
morabito,mario.versaci}@unirc.it

Abstract. Living cells possess properties that enable them to withstand the physiological environment as well as mechanical stimuli occurring within and outside the body. Any deviation from these properties will undermine the physical integrity of the cells as well as their biological functions. Thus, a quantitative study in single cell mechanics needs to be conducted. In this paper we will examine fluid flow and Neo-Hookean deformation. Particularly, a mechanical model to describe the cellular adhesion with detachment is proposed. Restricting the interest on the contact surface and elaborating again the computational results, it is possible to develop our idea about to reproduce the phases coexistence in the adhesion strip. Subsequently, a number of simulations have been carried out, involving a number of human cells with different mechanical properties. All the collected data have been used in order to train and test a suitable Artificial Neural Network (ANN) in order to classify the kind of cell. Obtained results assure good performances of the implemented classifier, with very interesting applications.

1 Introduction

In order to model and determine the effect of the blood flow in presence of a human cell, a Finite Element Method (FEM)-ANN-based approach has been exploited. FEM model requires the geometrical and physical definition of the blood vessel, the cell and flow parameters. For our purpose, we verify cell deformation under actual conditions. Thus, a brief description of the theoretical framework for the mechanical is given. Then, we describe the exploited approach through FEM analysis simulating the human cell and the blood flow, and so modeling an endothelial wall cross of a blood flow and the topological structure of the implemented ANN and finally, the obtained results. The goal of the paper was to focus the contact part among cell and endothelial wall about the deformation field. Our opinion is that the simulation notice the deformation inhomogeneity namely different concentration areas with different deformation values. This important observation should be connected with a specific form of the stored energy

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 720–727, 2009.

deformation that, in this case, loses the standard convexity to show the a non-monotone deformation law. Consequently, we have local minima and the variational problem seems more difficulty. Solutions through minimizing sequence are applied and this relieve microstructure formation. A number of simulations have been carried out, involving a number of human cells with different mechanical properties. All the collected data have been subsequently used in order to train and test a suitable ANN. The proposed ANN-approach has been exploited in order to classify the type of cell. Final results are satisfying: the approach is able to recognize the kind of cell.

2 Theoretical Approach: Problem Modeling

In this section of the paper we show how to simulate the effect of a blood flow in presence of a cell in a blood vessel. Using our FEM package [1,2,3], the blood flow and pressure drop across human cell have been studied and a mathematical model of the process has been constructed and analyzed. It consists of the equations of continuity (representing conservation of mass), motion (representing conservation of momentum) for the flow of blood through human cell [4] (see Fig. 1). These equations are supplemented by appropriate models which represent the stress/strain behavior of human cell [5]. Simulation of fluid-structure interaction is a challenging problem for computer modelers. In this work we exploited a FE approach to solve the developed mathematical 2D model representing the operations of human cell. 2D geometry represents a vertical section of the general 3D problem. According to our purposes, it represents a good trade-off between the analysis of stresses and the reduction of computational costs, thus assuring the achievement of our goals. For our aims, the Comsol Multiphysics® package has been exploited. Computer modeling can help in this context if it is based on an appropriate mathematical model and an accurate reliable solution. So it is needed to fit a model to the problem and get a satisfactory solution. Since the requirements about the length of the paper, please refers to [6] for the closed form of the analytic model.

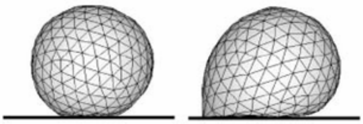


Fig. 1. 2D-view of cell deformation in presence of blood flow at t_1=0.125 (s)

3 FEM Approach

We modeled a horizontal flow channel (rectangular profile) in the middle of which there is a human cell (circular profile). For considering the deformation of the flow on the cell, we exploited the arbitrary Lagrangian-Eulerian technique [7,8,9].

The cell forces the fluid into a narrower path in the upper part of the channel, thus imposing a force on the structure's walls resulting from the viscous drag and fluid pressure. The cell structure, being made of a Neo-Hookean hyperelastic material, bends under the applied load. Consequently, the fluid flow also follows a new path, so solving the flow in the original geometry would generate incorrect results. The Navier-Stokes equations that solve the flow are formulated for these moving coordinates. The simulations exploit the FEM and require geometrical and physical definition of the blood flow and the human cell [10]; the latter has been modeled as a circumference with a portion of perimeter adherent to the venous paries. In this example the flow channel is 100 (μm) high and 300 (μm) long. The cell structure has a radius of 1.25 (μm), and is adherent 1 (μm) long at the channel's bottom boundary. Particularly, we propose the results for a red blood cell analysis. Assume that the structure is along the direction perpendicular to the image. The fluid is a water-like substance with a density ρ=1000 (kg/m^3) and dynamic viscosity η=0.001 (Pa·s). To demonstrate the desired techniques, assume the cell structure consists of a Neo-Hookean hyperelastic material with a density ρ=7850 (kg/m^3) initial tangent E=80 (kPa). The model consists of a fluid part, solved with the Navier-Stokes equations in the flow channel, and a structural mechanics part, which you solve in the human cell. Transient effects are taken into account in both the fluid and the cell structure. The structural deformations are modeled using large deformations in the Plane Strain application mode. The displacements and displacement velocities are denoted u, v, u_t, and v_t, respectively. Fluid flow is described by the Navier-Stokes equations for the velocity field, $\mathbf{u} = (u, v)$, and the pressure, p, in the spatial (deformed) moving coordinate system:

$$\begin{cases} \rho\frac{\partial \mathbf{u}}{\partial t} - \nabla\left(-\rho\mathbf{I} + \eta\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right)\right) + \rho\left((\mathbf{u}-\mathbf{u_m})\cdot\nabla\right)\mathbf{u} = \mathbf{F} \\ -\nabla\cdot\mathbf{u} = 0 \end{cases} \tag{1}$$

In these equations, $\mathbf{I}$ denotes the unit diagonal matrix, T the stress tensor and $\mathbf{F}$ is the volume force affecting the fluid. Assume that no gravitation or other volume forces affect the fluid, so that $\mathbf{F} = 0$. The Navier-Stokes equations are solved in the spatial (deformed) coordinate system. At the inlet, the model uses a fully developed laminar flow. Zero pressure is applied at the outlet. No-slip boundary conditions, that is $\mathbf{u} = 0$, are used at all other boundaries. Note that this is a valid condition only as long as you are solving the stationary problem. In this transient version of the same model, with the cell starting out from an undeformed state, it is necessary to state that the fluid flow velocity be the same as the velocity of the deforming obstacle. The coordinate system velocity is $\mathbf{u} = (u_m, v_m)$. At the channel entrance on the left, the flow has fully developed laminar characteristics with a parabolic velocity profile but its amplitude changes with time. At first, flow increases rapidly, reaching its peak value at 0.215 (s); thereafter it gradually decreases to a steady-state value of 3.33 (cm/s). The centerline velocity in the x direction, $\mathbf{u}_{in}$, with the steady-state amplitude U comes from the equation $\mathbf{u}_{in} = \frac{U\cdot t^2}{\sqrt{(0.04-t^2)^2+(0.1t)^2}}$, where t must be expressed in seconds. At the outflow (right-hand boundary), the condition is $p = 0$. On the solid (non deforming)

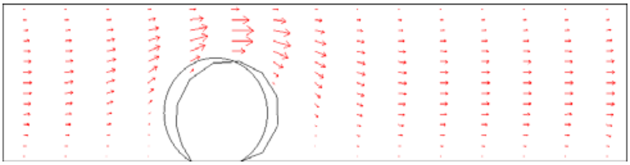


Fig. 2. Simulated flow and cell deformation at time instant t_0=0 (s) and t_1=0.215 (s)

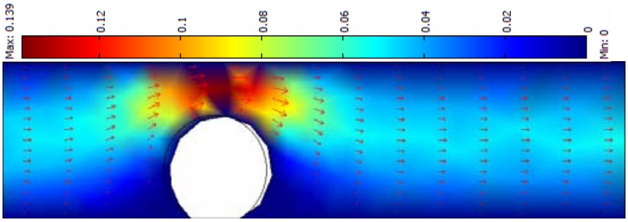


Fig. 3. Simulated flow velocity and geometry deformation at t=4 (s). The vectors indicate the flow direction and the color scale indicates flow-velocity magnitude (m/s).

walls, no-slip conditions are imposed, $u = 0$, $v = 0$, while on the deforming interface the velocities equal the deformation rate, $u_0 = u_t$ and $v_0 = v_t$. For boundary conditions, the cell is fixed to the bottom of the fluid channel, so that it cannot move in any direction. All other object boundaries experience a load from the fluid, given by $\mathbf{F}_T = -\mathbf{n} \cdot \left(-\rho\mathbf{I} + \eta\left(\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right)\right)$, where $\mathbf{n}$ is the normal vector to the boundary. This load represents a sum of pressure and viscous forces. With deformations of this magnitude, the changes in the fluid flow domain have a visible effect on the flow and on the cell, too (see Fig. 3). Fig. 3 shows the geometry deformation and flow at t=4 (s) when the system is close to its steady state. Due to the channel's small dimensions, the Reynolds number (R) of the flow is small ($R \ll 100$), and the flow stays laminar in most of the area. The swirls are restricted to a small area behind the structure. The amount of deformation as well as the size and location of the swirls depend on the magnitude of the inflow velocity. Fig. 4 further illustrates this point; it compares the average inflow velocity to the horizontal mesh velocity and the horizontal mesh displacement just beside the top of the structure at a generic physical point. For the fluid domain, the following settings are applied. For the boundary settings, we imposed a inlet condition with a mean velocity equal to u_{in} (set to 3.33 (cm/s)). For the sides we apply condition of wall with sliding absent; for the boundary ($\delta\Gamma$) on the right we imposed type of outlet with a condition of pressure and no viscous stress with $p_0 = 0$. The edges of the cell are characterized by conditions wall mobile dispersant (structural displacement) with the exception of the base cell which is fixed and not involved in the dynamic physical process, according to $\mathbf{n} \cdot \left(\eta_1\left(\nabla\mathbf{u_1} + (\nabla\mathbf{u_1})^T\right) - \rho\mathbf{I} - \eta_2\left(\nabla\mathbf{u_2} + (\nabla\mathbf{u_2})^T\right) - \rho_2\mathbf{I}\right) = 0$. Our studies have been based on a discrete domain with 231 elements. The number of

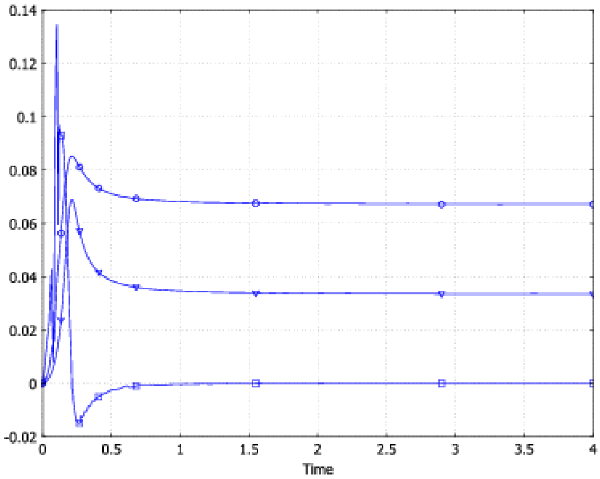
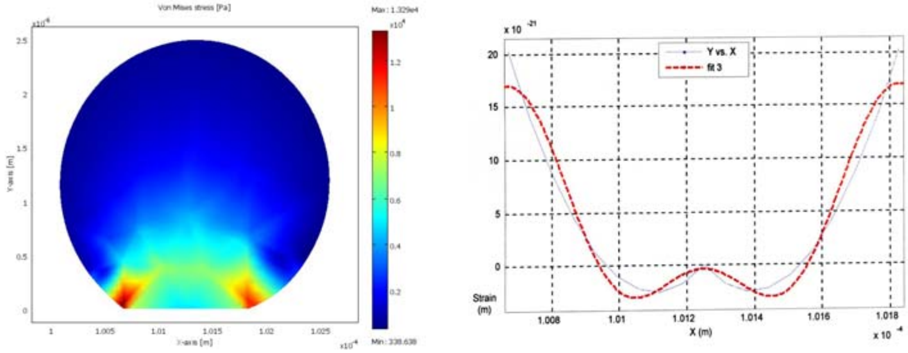


Fig. 4. Inflow velocity, horizontal mesh velocity, and mesh deformation. The curve with triangles shows the average x direction velocity at the inflow boundary (m/s); the curve with circles shows 104*mesh displacement in the x direction (m) at the generic geometry point; and the curve with squares shows 103*mesh velocity in the x direction (m/s), also at the same point.

degree of freedom is 1984. Mesh has been generated with triangular elements, having a geometric side of $5 * 10^{-4}$ (mm) for vessel and cell. We exploited a FEM implementation utilizing a time-dependent direct linear solver with parallel calculation [11]. Subsequently, we present final simulations in order to stress and strain results.

4 The Inverse Problem and Its Neural Network Based Regularization

The problem of estimate the kind of cell within the model starting from simulated measurements can be solved as a typical inverse problem of pattern recognition starting from available measurements. The proposed approach, useful to regularize the ill-posed problem and thus to discriminate the kind of cell, exploits Artificial Neural Network (ANN), particularly Multi-Layer Perceptron (MLP), trained with Levenberg-Marquardt method. Database used for our experimentations have been collected in our laboratory, by means of results obtained analyzing a number of human cells with the previously proposed FE based system. Human cells have the same dimensions, but with different mechanical properties. For each cell, a simulation has been carried out, in order to obtain the stress map according the specific properties. The collected database is composed by: 21 FEM maps equally distributed for the considered different kind of cells (i.e., keratocytes, leukocytes, red blood cells) and analyzed with different blood velocities; more other 21 FEM maps equally distributed for the considered different kind of



(a) Example map of the Von Mises stress in a red blood cell

(b) The non-monotone deformation law for a red blood cell

Fig. 5. Some results of the numerical modeling

cells, analyzed with the same blood velocity but with different mechanical and constitutive quantities. Then, in our experimentations, we considered only the areas depicting the adhesion strips. In order to set training signals, we made a trade-off between the requirements of an as large as possible training subset and a significant availability of testing signals. The training set has been composed by randomly selected sub-areas from each map of the collected database. On the other hand, the testing subset is represented by the whole adhesion strips. In this way, the so called inverse crime is avoided, since there are samples in the testing subset not included in training subset. Proposed computational intelligence application exploits an ANN as heuristic pattern reconstructor. ANN's inputs are: indices of cell mechanical properties (elastic modulus, viscosity), u_{in}, considering for the blood velocities, ρ and E; the average, standard deviation, skewness and kurtosis of the i-th point on computed map of the cell. The ANN's output is the kind of cell according to our database. The ANN based system was trained using Back-Propagation (BP) algorithm with adaptive rate of learning during a period of 600 epochs. The ANN, according to Kurková [12], has a hidden layer with 19 neurons; activation functions are: tan-sigmoid between input and hidden

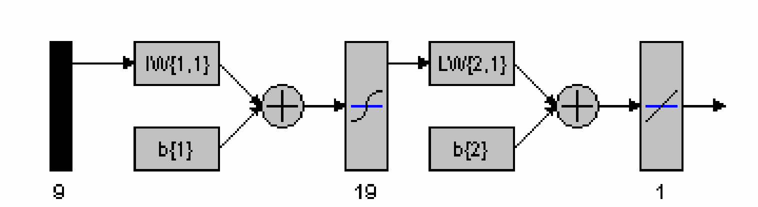


Fig. 6. Structure of the implemented ANN: $b\{1\}$ and $b\{2\}$ represent the biases of input and hidden layers respectively; $IW\{1,1\}$ and $LW\{2,1\}$ represent the weights for the input and the hidden layers respectively. 9 is the number of ANN-inputs (i.e., neurons in the input layer), 19 is the number of hidden neurons and 2 is the number of the ANN-outputs (i.e., output neurons)

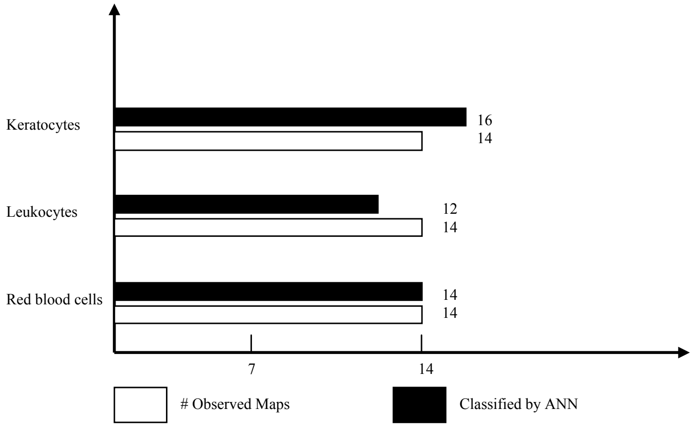


Fig. 7. Observed classes of map and results obtained by ANN

layer, and pure linear between hidden and output layer (see Fig. 6). After the training phase, the ANN has been tested; final results are shown in Fig. 7.

5 Conclusions

Proposed method provides a good overall accuracy, so our FEM-ANN package was very successfully in simulating fluid-structure interaction in the human blood vessel. Our interest was to point out the concentration or inhomogeneity of the deformation on the cell-wall contact area. This particular result open the way to simulate the adhesion-detachment problem through more sophisticated model (i.e. functional analysis tools) such that microstructural characterization can be emphasized. With these information, an ANN-based approach has been exploited in order to estimate the kind of human cell starting from signals obtained by computer simulations. Stress FEM maps have been used to train the ANN-based classifier. The proposed method provides a good overall accuracy in distinguishing the different kind of cell, as our experimentations demonstrated. At the same time, the procedure should be validate for cell with different shape and with different mechanical properties. Anyway, the presented results suggest the possibility of increasing and generalizing the performance of the ANN-based classifier; in this way it could be possible to refine its training step, for instance, including FEM's maps able to portrait cells with different spatial extension, making the training algorithm and finally the ANN more robust and flexible for other kind of cells. The authors are actually engaged in this direction.

Acknowledgments

Many thanks to doctor and scientists of Transplant Regional Center of Stam Cells and Cellular Therapy, "A. Neri", Reggio Calabria, Italy, for the availability and for medical support useful to write this paper.

References

1. COMSOL: FEMLAB User's guide Version 3.1, pp. 474–490 (2004)
2. Braess, D.: Finite Elements, 2nd edn. Cambridge University Press, Cambridge (2001)
3. Jin, J.: The Finite Element Method in Electromagnetics, 2nd edn. Wiley-IEEE Press, New York (2002)
4. N'Dri, N.A., Shy, W., Tran-Son-Tay, R.: Computational modeling of cell adhesion and movement using a continuum-kinetics approach. Biophysics Journal 85(4), 2273–2286 (2003)
5. Nassehi, V.: Practical Aspects of Finite Element Modelling of Polymer Processing. John Wiley and Sons, Chichester (2002)
6. Buonsanti, M., Cacciola, M., Megali, G., Morabito, F.C., Pellicanó, D., Versaci, M., Pontari, A.: Mechanical Aspects in the Cells Detachment. In: The 13th International Conference on Biomedical Engineering, ICBME 2008, Singapore (December 2008) (accepted)
7. Diacu, F.: An Introduction to Differential Equations. Freeman and Company, New York (2000)
8. Knupp, P., Margolin, L.G., Shashkov, M.: Reference Jacobian optimization-based rezone strategies for arbitrary Lagrangian Eulerian methods. Journal of Computational Physics 176(1), 93–128 (2002)
9. Glowinski, R., Pan, T.W., Hesla, T.I., Joseph, D.D.: A distributed lagrange multiplier/fictitious domain method for particulate flows. Int. J. Multiphase Flows 25, 201–233 (1998)
10. Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K., Watson, J.D.: Molecular biology of the cell, 3rd edn. Garland Science, Oxford (1994)
11. Dong, C., Lei, X.: Biomechanics of cell rolling: shear flow, cell-surface adhesion, and cell deformability. Journal of Biomechanics 33(1), 35–43 (2000)
12. Kurková, V.: Kolmogorov's theorem and multilayer neural networks. Neural Networks 5(3), 501–506 (1992)

Caller Interaction Classification: A Comparison of Real and Binary Coded GA-MLP Techniques

Pretesh B. Patel and Tshilidzi Marwala

School of Electrical and Information Engineering, University of the Witwatersrand,
Private Bag 3, 2050, Johannesburg, South Africa
p.patel@ee.wits.ac.za, t.marwala@ee.wits.ac.za
http://dept.ee.wits.ac.za/~marwala/

Abstract. This paper employs pattern classification methods for assisting contact centers in determining caller interaction at a 'Say account' field within an Interactive Voice Response application. Binary and real coded genetic algorithms (GAs) that employed normalized geometric ranking as well as tournament selection functions were utilized to optimize the Multi-Layer Perceptron neural network architecture. The binary coded genetic algorithm (GA) that used tournament selection function yielded the most optimal solution. However, this algorithm was not the most computationally efficient. This algorithm demonstrated acceptable repeatability abilities. The binary coded GA that used normalized geometric selection function yielded poor repeatability capabilities. GAs that employed normalized geometric ranking selection function were computationally efficient, but yielded solutions that were approximately equal. The real coded tournament selection function GA produced classifiers that were approximately 3% less accurate than the binary coded tournament selection function GA.

1 Introduction

This research focuses on a pattern classification problem utilized within an application that could assist contact centers in determining customer activities within their Interactive Voice Response (IVR) systems. During the last 5 years, the South African contact centre industry has experienced exceptional growth [1]. In order to gain a competitive advantage, contact centers are to fulfill customer expectations efficiently with informed responses and actions while discovering techniques to reduce overall cost of providing such a service [2]. It is therefore essential that contact centers evaluate the performance of their solutions in relation to customer interaction. This will assist in quantifying the self-service customer perception.

Customers want to resolve problems on their first call. They want convenient and reliable information fast. IVR systems can provide this. An IVR system is an automated telephony system that interacts with callers, gathers relevant information and routes calls to the appropriate destinations [2]. The inputs to the IVR system can be voice, Dual Tone Multi-Frequency (DTMF) keypad selection or a combination of the 2.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 728–735, 2009.

The aim of this research is to develop a field classification application, using computational intelligent methods, which could assist companies in quantifying customer activities within their IVR systems.

IVR applications are developed in Voice Extensible Markup Language (VXML). VXML applications are voice-based dialog scripts that consist of form or dialog elements. The form or dialog elements are used to group input and output sections together. A field element is used to obtain and interpret user input information. As a result, the form or dialog elements contain field elements [3].

The classification system developed categorizes caller behaviour at a field within the IVR applications into specific interaction classes. As a result, these interaction classes can assist in determining trends of caller behaviour within the self-service systems. For example, the field classification application can identify areas within the self-service applications that experienced the most caller disconnects. Thereafter, analysts can listen to a sample of these calls and determine the reason for this. The field classification system can also identify the fields that resulted in the majority of the callers transferring to a Customer Service Agent (CSA) due to difficulties experienced.

In order to develop such an application, the classification of data must be accurate. This paper details the development of artificial neural network (ANN) field classifiers using Multi- Layer Perceptron (MLP) neural network and genetic algorithms (GAs). GAs employing floating-point and binary representations are considered. Detailed explanations on these ANN architectures can be found in [4].

GAs are known to be robust optimization procedures based on the mechanism of the natural evolution. GAs have the capability of locating a global optimum as these procedures do not use any derivative information and GAs search from multiple points. In traditional GAs, binary representation has been used for chromosomes. Floating-point representation, real-coded GAs, of parameters as a chromosome has also been used [5].

The classification of data into various classes has been an important research area for many years. Artificial neural networks (ANNs) have been applied to pattern classification [6][7][8]. Fuzzy systems [9] and neural networks constructed using GAs have been utilized [10].

The section to follow provides a brief explanation of the field classification system. Thereafter, the implementation methodology is described. The paper ends with the comparison of the various field classifiers developed and the selection of the superior networks.

2 The Developed System

As the developed system is to be used to identify trends of caller behaviour at a field within the IVR VXML applications, the system is trained based on data extracted from IVR log event files. These files are generated by the IVR platform as specific events occur during a call to the system. Events such as call begin, form enter, form select, automatic speech recognition events, transfer events and call end events are captured in the logs [11].

Table 1. The inputs and outputs of the field classifier

Inputs	Outputs	Output interaction class
Confidence	Field performance	Good, acceptable, bad
No matches	Field transfer reason	Unknown, difficult
No inputs	Field hang-up reason	Unknown, difficult
Max speech timeouts	Field duration	High, medium, low
Barge-ins	Field recognition level	High, medium, low
Hang-ups		
Transfer to Customer Service Agent		
Duration		

Table 1 shows the inputs and outputs of the field classification system. These specific inputs have been selected to characterize the caller difficulty experienced at a field within a VXML application. The outputs of the classifiers summarize the caller field behaviour through the use of interaction classes. The confidence input illustrates the IVR speech recognition probability. The value is a percentage. A caller may answer a question the VXML application poses with a response the application did not anticipate. These events are represented by the no match inputs. Callers may reply to VXML applications by talking beyond the allocated timeout period of the field. These events correspond to the maximum speech timeout input parameters of the field classifiers [12]. When a question is presented by the automated application, a caller may remain silent. No input events represent these occurrences. In general, most self-service applications accommodate 3 of the above events per field. On the third attempt, if the caller is unsuccessful in completing the field, the caller is usually transferred to a CSA. These inputs are important as they assist in identifying difficulties experienced at the field.

The duration input parameter illustrates the time the caller consumed completing the VXML application field. The barge-in input parameter illustrates whether or not a caller interrupted the application while the automated question prompted played. Caller disconnects and transfers to Customer Service Agents (CSAs) are represented by the hang-up and transfer to agent input parameters, respectively. These inputs can also assist in determining the level of difficulty the caller experienced in the field.

The field performance output interaction class of the classifier will illustrate whether the caller behaviour is good, acceptable or bad. The field transfer reason and field hang-up reason interaction classes attempt to identify the motivation for the transfer to CSA or caller disconnect, respectively. The field duration as well as field recognition level classes illustrate 3 categories of performance, low, medium and high. As a result, these output parameters will assist in characterizing the caller experience at a VXML field.

Once the field classification system has been employed to assess the performance of various interaction areas within the VXML applications, the caller perception of the solution can be determined. For example, after analyzing the outputs of the classifiers, it is determined many callers are disconnecting due to difficulties experienced during the preliminary fields of an application, it can be concluded that the callers are frustrated with the system. In order to reduce the caller dissatisfaction, it would be essential to optimize the various grammars that interpret the caller responses.

3 Implementation Methodology

The development process was divided into various stages. The remainder of this section will elaborate on these stages of implementation The following procedure has been pursued in the creation of the various classifiers employed:

1. Selection and pre-processing of data to be used by the classifiers.
2. Optimization of the classifier architectures using GAs.
3. Comparison of the various GAs developed and the selection of the superior network.

3.1 Selection and Pre-processing of Data

The data utilized in developing the ANNs is based on data extracted from IVR log event files. A parsing application that extracted information such as recognition confidence values, caller barge-in information has been utilized to generate the data sets. This application also executed specific instructions in the creation of the data sets. Rules such as if 3 occurrences of no inputs, no matches or maximum speech timeouts occur within the caller interaction to a particular field and a transfer occurs thereafter, the field transfer reason computed will be 'difficulties' were followed.

In order to present the no match, no input and maximum speech timeout information to the field classifiers, a binary notation has been employed. These inputs are presented by 3 digit binary words. For example, if a no match 1 and a no match 2 occur at a field, the binary notation will be '011'. A similar binary notation is employed for the no input and maximum speech timeout classifier inputs. The barge-in, hang-up and transfer to CSA input information were represented by bit binary words. A similar binary notation scheme has also been utilized to interpret the interaction classes outputted. The data is divided into training, validation and test sets. The validation data set is used to assess the network and the test data is used to confirm the classification capability of the developed networks.

The confidence and duration input parameters of the classifiers were preconditioned by normalizing the data. Due to the binary word representation utilized to present the remaining inputs, normalization of these input parameters is not necessary.

3.2 Optimization of Classifier Architecture Using Genetic Algorithms

This stage of implementation involved the optimization of the ANN architectures. As a result, this step of development involved the identification of the correct number of hidden neurons that would yield the most accurate results.

Binary and real coded GAs were employed to optimize the field classifier architecture. Populations of MLP ANNs were generated by the GAs. Due to the MLP ANN non-linear capabilities, they are said to be excellent universal approximators that provide highly accurate solutions. As a result, these networks produce very practical tools for classification and inversion problems [4].

It has been stated that a network with 1 hidden layer, provided with sufficient data, can be used to model any function [4]. Therefore, the MLP ANNs employed consisted of only 1 hidden layer. The MLP ANN hidden layer consists of non-linear activation functions. The choice of the activation function is largely dependent on the application of the model [4]. However, it has been found that the hyperbolic tangent activation function offers a practical advantage of faster convergence during training [6]. As a result, this function has been employed within the MLP network. The most appropriate selection of the output layer activation function for a classification problem is the logistic sigmoidal function [6]. Therefore, this function has been employed within the output layer of the MLP network.

An error function that mapped the number of hidden nodes to the accuracy of the developed network was used as the evaluation function for the GAs. The fitness of the individuals within a population was determined by calculating the accuracy of the ANNs when presented with validation and test data sets. The minimum value of these accuracies determined the fitness of the individual.The outputs of the ANNs were interpreted by utilizing a classification threshold value of 0.5. This value proved to be adequate for the MLP ANN implementations. A confusion matrix is utilized to identify the number of true and false classifications that are generated by the models developed. This is then used to calculate the true accuracy of the classifiers, using the following equation:

$$Accuracy = \sqrt{\frac{TP * TN}{(TP + FN) * (FP + TN)}} \tag{1}$$

where
TP is the true positive (1 classified as a 1),
TN is the true negative (0 classified as a 0),
FN is the false negative (1 classified as a 0),
FP is the false positive (0 classified as a 1).

The GAs produced 25 generations of 10 MLP ANN individuals within the population. The GAs were limited to produce MLP ANN individuals with the number of hidden nodes between 5 and 100. Networks with hidden nodes greater than 100 were not developed due to the generalization capabilities reducing as the number of intermediate units increase [13].

In order to produce successive generations, the selection function determines which of the individuals will survive to the next generation. Roulette wheel selection, scaling techniques, tournament, normal geometric, elitist models and ranking methods are examples of selection functions used [5]. The selection approach assigns a probability of selection to each individuals based on its fitness value. This research compares solutions produced by GAs that employ normalized geometric ranking and tournament selection functions.

3.3 Comparison of the Various Genetic Algorithms and Selection of the Superior Model

The binary and real coded GAs were compared in terms of their repeatability, computational efficiency as well as quality of the solution.

Repeatability is defined as the number of repetitions that returned the same optimal number of hidden nodes. Repeatability has been demonstrated by executing 8 repetitions of the binary and real coded GAs. This investigation revealed that the binary coded ranking selection GA returned different number of hidden nodes with different fitness values on each execution of the algorithm. Figure 1 also illustrates that the remaining GAs returned the same number of hidden nodes with the same fitness value for a number of the repetitions.

Computational efficiency, in this context, is defined as the number of generations the GA utilized to converge to the most optimal number of hidden nodes. Figure 1 illustrates that, the real coded tournament selection GA is most efficient. In the majority of the repetitions, this GA converged to an optimal solution before 9 generations. In the majority of the investigations, the binary coded ranking and the binary coded tournament selection GA converged to an optimal solution before 10 generations. However, the real coded ranking selection GA, in majority of the repetitions, converged before generation 13. Figure 1 also illustrates that the binary coded tournament selection GA resulted in the most accurate MLP ANNs with accuracies of approximately 96.45%. This algorithm returned the same fitness value for 2 of the 8 repetitions. The number of hidden nodes corresponding to this accuracy is 5. However, the algorithm converged to this solution at generation 23.

Quality of the GA solution is the verification that the number of hidden nodes returned by the GA is really the most optimal value. It has been determined that 5, 6, 7 and 9 number of hidden nodes resulted in the most accurate MLP classifiers. This has been determined by identifying the most accurate MLP classifiers generated from the 8 GA repetitions executed. These number of hidden nodes were identified by the binary coded tournament, real coded ranking, binary coded ranking and real coded tournament selection GAs, respectively. In order to determine the quality of the GA solutions, MLP classifiers were created containing these number of hidden nodes. The accuracy of these networks was determined by using equation 1. The sensitivity and false positive ratio were also calculated to measure the true performance of the networks. Table 2 illustrates the results of the quality of solution investigation. As illustrated, the number of hidden nodes that resulted in the most accurate MLP classifier is 5. This number of hidden nodes creates a network that performs accurately on both the validation and test data sets. As a result, the classifier has good generalization capabilities, as compared to the other MLP classifiers that perform approximately 4% more accurately on the test data set. The sensitivity and false positive ratio values also support these findings.

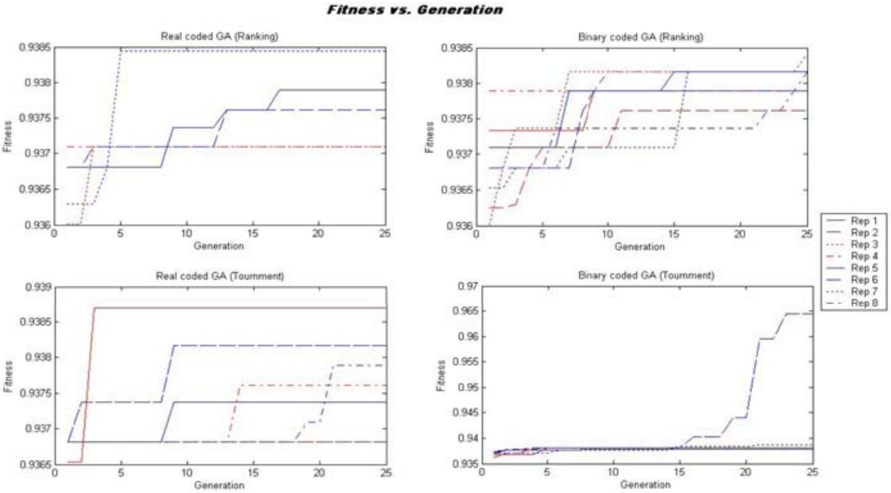


Fig. 1. This figure shows the Fitness vs. Generation for all 8 repetitions

Table 2. This table illustrates the various models that were created. Accuracies are presented as percentages.

Hidden Nodes	Accuracy (Validation)	Accuracy (Test)	Hit Rate (Validation)	Hit Rate (Test)	False Alarm Rate (Validation)	False Alarm Rate (Test)
5	96.36	96.80	95.02	95.12	0.023	0.015
6	93.73	98.17	90.12	97.08	0.025	0.007
7	93.68	98.00	89.98	96.75	0.025	0.007
9	93.60	97.98	89.83	96.74	0.025	0.008

4 Conclusion

This research entailed the development of a 'Say account' field classification system. MLP networks were used to classify caller interactions. Binary coded and real coded GAs that utilized ranking as well as tournament selection functions were also employed to optimize the classifier architecture.

The development methodology utilized for creating all the networks involved, initially, pre-processing the data sets. This ensured that the classifiers would interpret the inputs proficiently. Thereafter, the numbers of hidden nodes were optimized utilizing the GA algorithm. This resulted in creating acceptable network architecture. GA results were compared in terms of computational efficient, repeatability and the quality of the solution. These analyses assisted in determining the algorithm that was most suited to this application and the algorithm that yielded the most accurate classifier. Acceptable classification accuracies were achieved. The most accurate network that illustrated excellent generalization capabilities yielded accuracies of approximately 96%. This illustrates that MLP ANNs are proficient in classifying field caller interaction.

The binary coded GA that utilized tournament selection yielded the most accurate MLP classifier. The algorithm yielded the same result on 2 of the 8 repetitions. The number of hidden nodes of 5 returned by the algorithm was confirmed to be the optimal in the quality of the solution investigation. However, the algorithm converged at this solution at generation 23 of 25 generations. As a result, it can be concluded that this GA is most suited to this application in terms of optimal solution, but it is not the most computational efficient algorithm.

References

1. The first major South African Contact Centre and BPO Market Quantification study in over 5 years, http://www.contactindustryhub.co.za/research.php (last accessed August 17, 2008)
2. Nichols, C.: The Move from IVR to Speech – Why This is the Right Time to Make the Move to Speech Applications in Customer-Facing Operations. Intervoice (2006)
3. VoiceXML 2.0/VoiceXML 2.1 Reference, http://developer.voicegenie.com/voicexml2tagref.php (last accessed August 17, 2008)
4. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
5. Houck, C.R., Joines, J.A., Kay, M.G.: A genetic algorithm for function optimization: a Matlab implementation. NCSU-IE Technical Report. North Carolina State University (1995)
6. Nabney, I.T.: Netlab: Algorithms for Pattern Recognition. Springer, Heidelberg (2002)
7. Patel, P.B., Marwala, T.: Neural Networks, Fuzzy Inference Systems and Adaptive-Neuro Fuzzy Inference Systems for Financial Decision Making. In: King, I., Wang, J., Chan, L.-W., Wang, D. (eds.) ICONIP 2006. LNCS, vol. 4234, pp. 430–439. Springer, Heidelberg (2006)
8. Marwala, T.: Fault classification using pseudo-model energies and neural networks. American Institute of Aeronautics and Astronautics 41, 82–89 (2003)
9. Russo, M.: FuGeNeSys A fuzzy genetic neural system for fuzzy modeling. IEEE Transaction on Fuzzy Systems 6, 373–388 (1998)
10. Abdella, M., Marwala, T.: The use of genetic algorithms and neural networks to approximate missing data in database. Computing and Informatics 24, 1001–1013 (2006)
11. VoiceGenie Technologies Inc. VoiceGenie 7 Tools Users Guide. VoiceGenie Technologies Inc. (2005)
12. VoiceXML properties, http://community.voxeo.com/vxml/docs/nuance20/VXMLproperties.html (last accessed August 25, 2008)
13. Baum, B.E., Haussler, D.: What size net gives valid generalization? Neural Computation 1, 81–90 (1989)

A Robust Technique for Background Subtraction in Traffic Video

Tao Gao[1], Zheng-guang Liu[1], Wen-chun Gao[2], and Jun Zhang[1]

[1] School of Electrical Engineering and Automation, Tianjin University,
Tianjin, 300072, China
[2] Honeywell (China) Limited, Tianjin, 300042, China
gaotao231@yahoo.cn

Abstract. A novel background model based on Marr wavelet kernel and a background subtraction technique based on binary discrete wavelet transforms are introduced. The background model keeps a sample of intensity values for each pixel in the image and uses this sample to estimate the probability density function of the pixel intensity. The density function is estimated using a new Marr wavelet kernel density estimation technique. Since this approach is quite general, the model can approximate any distribution for the pixel intensity without any assumptions about the underlying distribution shape. The background and current frame are transformed in the binary discrete wavelet domain, and background subtraction is performed in each sub-band. Experiments show that the simple method produces good results with much lower computational complexity and can effectively extract the moving objects, even though the objects are similar to the background, thus good moving objects segmentation can be obtained.

1 Introduction

Identifying moving objects from a video sequence is a fundamental and critical task in many computer-vision applications. Background subtraction [1, 2] is a method typically used to detect unusual motion in the scene by comparing each new frame to a model of the scene background. In traffic video surveillance systems, stationary cameras are typically used to monitor activities on the road. Since the cameras are stationary, the detection of moving objects can be achieved by comparing each new frame with a representation of the scene background. This process is called background subtraction and the scene representation is called the background model. Typically, background subtraction forms the first stage in automated visual surveillance systems. Results from background subtraction are used for further processing, such as tracking targets and understanding events. A mixture of Gaussians is used in [3] to model each pixel's intensity. The models are learned and updated for each pixel separately. A mixture of three normal distributions is used in [4] to model the pixel value for traffic surveillance applications. The pixel intensity is modeled as a weighted mixture of three normal distributions: road, shadow and vehicle distribution. An incremental EM algorithm is used to learn and update the parameters of the model. In [5, 6] kalman and wiener filtering are performed at every pixel. Foreground

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 736–744, 2009.

is detected when the observed intensity is different than the predicted intensity. Ref. [7] modifies the background image that is subtracted from the current image so that it looks similar to the background in the current video frame. The background is updated by taking a weighted average of the current background and the current frame of the video sequence. In this paper, we propose a new model for background maintenance and subtraction. A sample of intensity values for each pixel is kept and used to estimate the Marr wavelet probability density function of the pixel intensity. After background modeling, binary discrete wavelet transforms is used for background subtraction. In our experiments about the video surveillance on urban road, the model can solve the problem of gradual change of illumination and it can detect both moving vehicles and foot passengers.

2 Previous Background Modeling Methods

Background modeling is at the heart of any background subtraction algorithm. Several models have been put forward for background maintenance and subtraction described in introduction. In this paper, we focus only on the two most commonly used techniques, and exclude those which require significant resource for initialization or are too complex.

2.1 Frame-Difference Background Modeling

Frame-difference method [7, 8, 9] obtains the background image as follows:

$$BW_i = \begin{cases} 1 & if \quad abs(I_i - I_{i-1}) \geq A \\ 0 & if \quad abs(I_i - I_{i-1}) < A \end{cases} \tag{1}$$

The histogram of the difference image will have high values for low pixel intensities and low values for the higher pixel intensities. To set the threshold A, a dip is looked for in the histogram that occurs to the right of the peak. Starting from the pixel value corresponding to the peak of the histogram, we search toward increasing pixel intensities for a location on the histogram that has a value significantly lower than the peak value (using 10% of the peak value). The corresponding pixel value is used as the new threshold A. The background B_i:

$$B_i = \begin{cases} B_{i-1}(x,y) & BW_i(x,y) = 1 \\ aI_i + (1-a)B_{i-1}(x,y) & BW_i(x,y) = 0 \end{cases} \tag{2}$$

Then B_i is the background image. The weight a assigned to the current and instantaneous background affect the update speed, empirically determined to be 0.1.

2.2 Mixture of Gaussians Background Modeling

In Mixture of Gaussians (MoG) [10], each pixel location is represented by a number (or mixture) of Gaussians functions that sum together to form a probability distribution function F :

$$F(i_t = \mu) = \sum_{i=1}^{k} \omega_{i,t} \cdot \eta(\mu, \sigma) \tag{3}$$

To determine if a pixel is part of the background, we compare the input pixels to the means μ_i of their associated components. If a pixel value is close enough to a given component's mean, that component is considered a matched component. Specifically, to be a matched component, the absolute difference between the pixel and mean must be less than the component's standard deviation scaled by a factor $D: \left| i_i - \mu_{i,t-1} \right| \leq D \cdot \sigma$. Then we update the component variables (ω, μ, and σ) to reflect the new pixel value. For matched components, a set of equations increase our confidence in the component (ω increases, σ decreases, and μ is nudged towards the pixel value). For non-matched components, the weights decrease exponentially (μ and σ stay the same). How fast these variables change is dependent on a learning factor p present in all the equations. Then we determine which components are parts of the background model. First, we order the components according to a confidence metric ω / σ, which rewards high ω and low σ. We do this because we want to keep only the M most confident guesses. Second, we apply a threshold to the component weights ω. The background model is then the first M components (in order of highest to lowest ω / σ), whose weight ω is above the threshold. M is the maximum number of components in the background model, and reflects the number of modes we expect in the background probability distribution function F. Last, we determine foreground pixels. Foreground pixels are those that don't match any components determined to be in the background model.

3 A Robust Background Modeling Method

Marr wavelet [11] is the second derivative of Gaussian smooth function $\psi_m = C_m (-1)^m \frac{\mathrm{d}^2}{\mathrm{d}t^m} (\mathrm{e}^{-\frac{t^2}{2}}) |_{m=2}$, that is:

$$\psi(t) = \frac{2}{\sqrt{3}} \pi^{-1/4} (1 - t^2) \mathrm{e}^{-t^2/2} \tag{4}$$

The coefficient $\frac{2}{\sqrt{3}} \pi^{-1/4}$ is a guarantee of normalization of $\psi(t)$: $\|\psi\|^2 = 1$. Marr wavelet is widely used in visual information processing, edge detection and other fields. The difference of Gaussians (DOG) is a good approximation to Marr wavelet. In practice, we use DOG to approximate Marr wavelet:

$$\psi(t) = \mathrm{e}^{-\frac{t^2}{2}} - \frac{1}{2} \mathrm{e}^{-t^2/8} \tag{5}$$

The first frame can be set as the initial background. Considering the dithering of camera, the bias matrix should be obtained before background modeling. We mainly concern the up-down, left-right plane dithering. If the background is B and current frame is f , they are firstly processed with two or three level Gaussian pyramid decomposition:

$$f_{n-1} = (f_n \otimes \frac{e^{-(n_1^2+n_2^2)/(2\sigma^2)}}{\sum_{n_1}\sum_{n_2} e^{-(n_1^2+n_2^2)/(2\sigma^2)}}) \downarrow_2 \tag{6}$$

Where $n_1 = n_2$ =3, and σ is 0.5. Supposing the bias matrix is $[v,h]$, v and h representative the vertical and level dithering parameters. If the size of f_{n-1} is $M \times N$, the n-1 level initial bias matrix is $[v_{n-1}, h_{n-1}]$, and the maximum pixel value is $f_{\max}$, minimum is $f_{\min}$, so the bias function is as follows:

$$\begin{aligned} Mo = \frac{\alpha}{\sqrt{2\pi\sigma}} \mid & e^{-(f_{n-1}-B_{n-1,v_{n-1},h_{n-1}}(\Delta x,\Delta y))^2/2\sigma} \\ & -\frac{1}{2}e^{-(f_{n-1}-B_{n-1,v_{n-1},h_{n-1}}(\Delta x,\Delta y))^2/8\sigma} \mid +(1-\alpha)\frac{1}{f_{\max}-f_{\min}} \end{aligned} \tag{7}$$

Where α is 0.3, and σ is $\beta \cdot (f_{\max} - f_{\min})^2$, β=0.01, $(\Delta x, \Delta y)$ is the offset distance. The $(\Delta x, \Delta y)$ is changed to obtain the maximum value Ds according to $Ds = \log(Mo) / \sum_M \sum_N f_{n-1}$, and thus to get the best offset distance $(\Delta x_{best}, \Delta y_{best})$. The $[v_{n-1} + \Delta x_{best}, h_{n-1} + \Delta y_{best}]$ is used as the initial bias matrix for next level f_n . By the iterative process, the dithering distance can be finally determined. If the original background after offset correction is $B(i,j)$, and current frame is $f(i,j)$, we define the probability distribution of deviation between background and current frame is:

$$PS(i,j) = \frac{\alpha}{\sqrt{2\pi\sigma}} \left| e^{-\frac{(B(i,j)-f(i,j))^2}{2\sigma}} - \frac{1}{2} e^{-\frac{(B(i,j)-f(i,j))^2}{8\sigma}} \right| \tag{8}$$

The updating weight for background pixel is as follows:

$$\Delta S(i,j) = \frac{PS(i,j)}{(PS(i,j) + (1-\alpha)\frac{1}{f_{\max} - f_{\min}})} \tag{9}$$

The iterative process for updating background is:

$$B_i = \frac{\sum_{n=1}^{N} \Delta S_n \cdot (B_{i-1} - f_n)^2}{\sum_{n=1}^{N} \Delta S_n} \tag{10}$$

i is the iteration number, N is the frame number. Then B_i is the final background.

4 BDWT Based Background Subtraction

One dimension signal $f(t)$, the binary discrete wavelet [12] transforms are as follows:

$$S_{2^j} f(t) = f(t) * \phi_{2^j}(t) \tag{11}$$

$$W_{2^j} f(t) = f(t) * \varphi_{2^j}(t) \tag{12}$$

$S_{2^j} f(t)$ is the projection of $f(t)$ in the V_j space, and $W_{2^j} f(t)$ is the projection of $f(t)$ in the W_j space. In frequency domain:

$$S_{2^j} \hat{f}(\omega) = \hat{f}(\omega) \cdot \hat{\phi}(2^j \omega) = S_{2^{j-1}} \hat{f}(\omega) H(2^{j-1} \omega) \tag{13}$$

$$W_{2^j} \hat{f}(\omega) = \hat{f}(\omega) \cdot \hat{\varphi}(2^j \omega) = S_{2^{j-1}} \hat{f}(\omega) G(2^{j-1} \omega) \tag{14}$$

So formula (11) (12) can be rewritten as:

$$S_{2^j} f(t) = \sum_{l \in Z} S_{2^{j-1}} f(t-l) h_{j-1}(l) \tag{15}$$

$$W_{2^j} f(t) = \sum_{l \in Z} S_{2^{j-1}} f(t-l) g_{j-1}(l) \tag{16}$$

For a digital signal $d(n) = S_1 f(t)\,|_{t=nT} = S_1 f(n)$,

$$S_{2^j} f(n) = \sum_{l \in Z} S_{2^{j-1}} f(n-l) h_{j-1}(l) \quad j = 1, 2, \cdots \tag{17}$$

$$W_{2^j} f(n) = \sum_{l \in Z} S_{2^{j-1}} f(n-l) g_{j-1}(l) \quad j = 1, 2, \cdots \tag{18}$$

Because the coefficients of the sub-bands of the BDWT are highly correlated, and the direction and size are the same as the image, also, there is no translation in the sub-bands; we define the difference between digital signal $d_1(n)$ and $d_2(n)$ as:

$$DE = \sum_{j=J_0}^{J_1} \{ \left| S_{2^j} f_1(n) - S_{2^j} f_2(n) \right| + \left| W_{2^j} f_1(n) - W_{2^j} f_2(n) \right| \} \tag{19}$$

The J_0 and J_1 are the starting and ending scales. For a two dimensions digital image, we perform the BDWT to the rows and then the columns. After acquiring the DE between two frames, the motion area is obtained by setting a threshold which can be obtained automatically by otsu [13] method. The noise can be removed by mathematical morphology method. In application, one of the two frames is a background image; the other is the current frame.

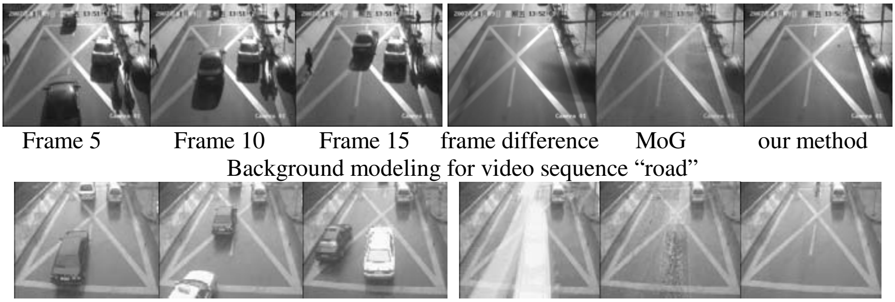
5 Experimental Results

In this section, we compare the performance of some background modeling techniques: frame-difference, MoG, and our method with uses BDWT based motion segmentation to perform background subtraction. The video sequences used for testing were taken from urban traffic monitoring cameras. The video was sampled at a resolution of 768x576 and a rate of 15 frames per second. Once the frames were loaded into memory, our algorithm averaged 18 frames per second on a 1400 MHz Celeron CPU. Although we used grayscale video as input, and output a grayscale background model, it is straightforward to extend the algorithm to use color images.

We use the error measurement quantify how well each algorithm matches the ground-truth. It is defined in our context as follows:

$$error = \frac{\left|\text{Number of foreground pixels identied by the algorithm - Number of foreground pixels in ground-truth}\right|}{\text{Number of foreground pixels in ground-truth}}$$

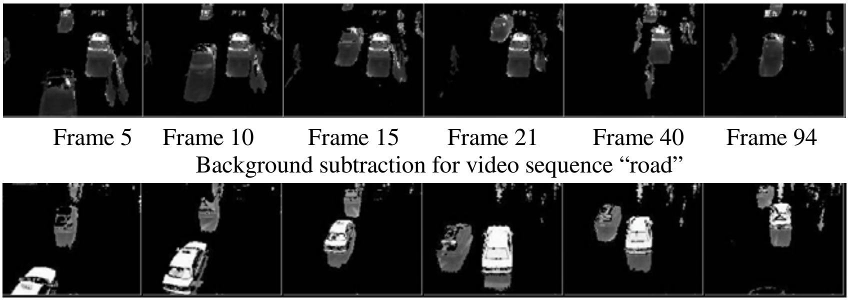
Figure.1 shows three sample frames of each test video, and the background modeling results for every video sequence. The frame number of video “road” and “car” is respectively 160 and 224. Figure.2 shows the background subtraction by our method. Figure.3 shows background subtraction by frame-difference, and Figure.4 shows background subtraction by MoG. Figure.5 and Figure.6 shows the error measurement of each method for video sequence “road” and “car”. From the experiments we can see that the frame-difference method has two drawbacks: first, as the foreground objects may have a similar color as the background, these objects can not be detected by threshold. Second, the method is only slowly adapting to slightly changing environmental conditions. Thus, faster changes as a flashlight signal or fluttering leaves in the wind can not be modeled. Also, in practice, modeling the background variations with a small number of Gaussian distribution will not be accurate. Furthermore, the very wide background distribution will result in poor detection because most of the gray level spectrum would be covered by the background model. Experiments show that our method is more robust against changes in illumination. While some complicated techniques can also produce superior performance, experiments show that our simple method can produce good results with much lower computational complexity.



Background modeling for video sequence “road”

Background modeling for video sequence “road”

Fig.1. The comparison of background modeling



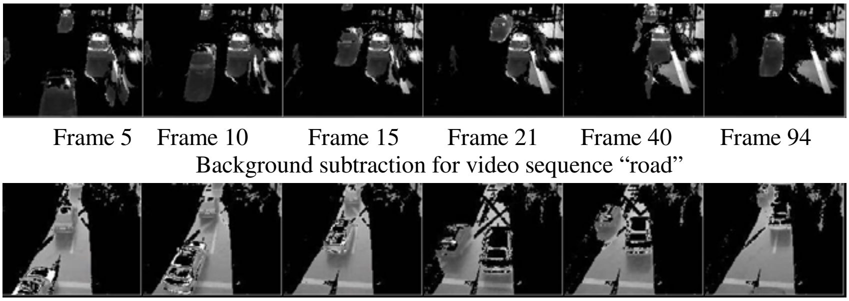
Frame 5 Frame 10 Frame 15 Frame 21 Frame 40 Frame 94

Background subtraction for video sequence “road”

Frame 51 Frame 64 Frame 81 Frame 172 Frame 186 Frame 212

Background subtraction for video sequence “car”

Fig. 2. Background subtraction by our method



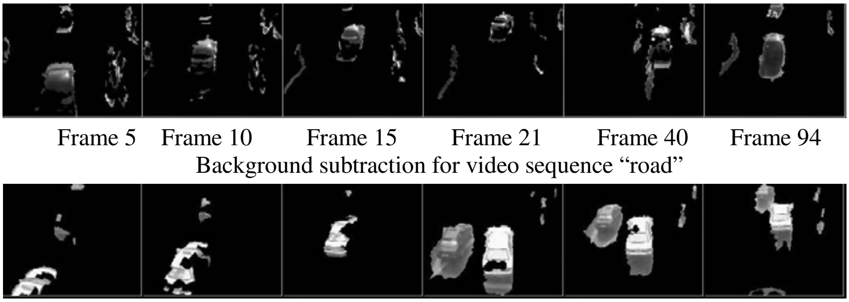
Frame 5 Frame 10 Frame 15 Frame 21 Frame 40 Frame 94

Background subtraction for video sequence “road”

Frame 51 Frame 64 Frame 81 Frame 172 Frame 186 Frame 212

Background subtraction for video sequence “car”

Fig. 3. Background subtraction by frame-difference



Frame 5 Frame 10 Frame 15 Frame 21 Frame 40 Frame 94

Background subtraction for video sequence “road”

Frame 51 Frame 64 Frame 81 Frame 172 Frame 186 Frame 212

Background subtraction for video sequence “car”

Fig. 4. Background subtraction by MoG

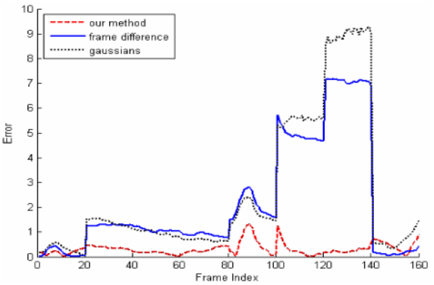


Fig. 5. Error measurement of sequence "road"

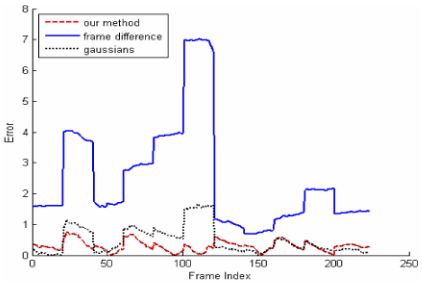


Fig. 6. Error measurement of sequence "car"

6 Conclusion

In this paper, we introduced a novel background model and a background subtraction technique based on wavelet theory. The model keeps a sample of intensity values for each pixel in the image and uses this sample to estimate the Marr wavelet probability density function of the pixel intensity. The density function is estimated using Marr wavelet kernel density estimation technique. Background subtraction is based on binary discrete wavelet transforms. Experimental results show that our method is robust against environmental noise and illumination change, and can segment the whole parts of foreground successfully.

Acknowledgments. The research is funded by Tianjin traffic bureau, China, and also funded by Tianjin Subway Safety System: [3240120], **Honeywell** (China) Limited.

References

1. Grimson, W., Stauffer, C., Romano, R., Lee, L.: Using Adaptive Tracking to Classify and Monitor Activities in a Site. In: IEEE Conf. on Computer Vision and Pattern Recognition, pp. 22–29 (1998)
2. Sen-Ching, S., Cheung, Kamath, C.: Robust Techniques for Background Subtraction in Urban Traffic Video. In: Proc. SPIE Visual Communications and Image Processing, pp. 881–892 (2004)
3. Stauffer, C., Grimson, W.E.L.: Adaptive Background Mixture Models for Real Time Tracking. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pp. 246–252 (1999)
4. Friedman, N., Russell, S.: Image Segmentation in Video Sequences: A Proba-bilistic Approach. In: Proc. Thirteenth Conf. on Uncertainty in Artificial Intelligence (1997)
5. Koller, D., Weber, J., Huang, T., et al.: Towards Robust Automatic Traffic Scene Analysis in Real-Time. In: Proc. Int. Conf. on Pattern Recognition, pp. 126–131 (1994)
6. Toyama, K., Krumm, J., Brumitt, B., Meyers, B.: Wallflower: Principles and Practice of Background Maintenance. In: Proc. IEEE Int. Conf. on Computer Vision, pp. 255–261 (1999)
7. Gupte, S., Masoud, O., Martin, R.F.K., et al.: Detection and Classification of Vehicles. IEEE Transactions on Intelligent Transportation Systems 3(1), 37–47 (2002)

8. Wang, W., Sun, M.: A New Algorithm for Detecting Object Under Traffic Scene. Computer Applications and Software 22(4), 90–92 (2005)
9. Guo, Y.T., Song, H.S., He, Y.Y.: Algorithms for Background Extraction in Video Traffic Monitoring System. Video Engineering (5), 91–93 (2006)
10. Stauer, C., Grimson, W.: Learning Patterns of Activity Using Real-Time Tracking. IEEE Trans. on Pattern Analysis and Machine Intelligence 22, 747–757 (2000)
11. Zhang, X.G., Ding, X.H.: A Note on Continuous Wavelet Transform. Journal of Guangxi Academy of Sciences 23(1), 4–6 (2007)
12. Swanson, M.D., Tewfik, A.H.: A Binary Wavelet Decomposition of Binary Images. IEEE Transactions on Image Processing 5(12), 1637–1650 (1996)
13. Otsu, N.: A Threshold Selection Method from Gray-Level Histogram. IEEE Trans. SMC 9(1), 62–66 (1979)

Gabor Filters as Feature Images for Covariance Matrix on Texture Classification Problem

Jing Yi Tou[1], Yong Haur Tay[1], and Phooi Yee Lau[1,2]

[1] Computer Vision and Inteligent Systems (CVIS) group,
Universiti Tunku Abdul Rahman (UTAR).
9, Jalan Bersatu 13/4, 46200 Petaling Jaya, Selangor, Malaysia
tayyh@utar.edu.my
[2] Instituto de Telecomunicações. Av. Rovisco Pais, 1049-001 Lisboa, Portugal
laupy@lx.it.pt

Abstract. The two groups of popularly used texture analysis techniques for classification problems are the statistical and signal processing methods. In this paper, we propose to use a signal processing method, the Gabor filters to produce the feature images, and a statistical method, the covariance matrix to produce a set of features which show the statistical information of frequency domain. The experiments are conducted on 32 textures from the Brodatz texture dataset. The result that is obtained for the use of 24 Gabor filters to generate a 24 × 24 covariance matrix is 91.86%. The experiment results show that the use of Gabor filters as the feature image is better than the use of edge information and co-occurrence matrices.

1 Introduction

Texture classification has been studied for years because of its usefulness in many computer vision applications. In these applications, the texture analysis helps to recognize the images through its texture information [1], such as wood species recognition [2][3], rock classification [4], face detection [5] and etc.

The texture classification methods can be divided into five main groups in general, namely the; 1) structural; 2) statistical; 3) signal processing; 4) model-based stochastic [1] and; 5) morphology-based methods [6]. The statistical and signal processing methods are most widely used because they can be used on general textures while other methods have more restriction on the characteristics of the textures that they can be implemented on. Some of these methods can be combined for better performance.

In our previous work [7], a combination of a statistical method, the grey level co-occurrence matrices (GLCM) and a signal processing method, the Gabor filters is used. The combination here is to append both the GLCM and Gabor features as a single feature vector. In this paper, we propose a combination method where the signal processing method, the Gabor filters are used as the feature images to generate the covariance matrix which is a statistical method.

Section 2 shows the Gabor filters and the covariance matrix algorithms used in the paper. Section 3 shows the dataset and settings used in the experiments conducted. Section 4 shows the experiment results and analysis. Section 5 shows the conclusion and future works.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 745–751, 2009.

2 Gabor Filters and Covariance Matrix

The Gabor filters is a type of signal processing method while the covariance matrix is a statistical method. In this paper, the Gabor filters are used as feature images, which are images or two-dimensional matrices produced by a feature extraction algorithm, that are used to generate a covariance matrix.

2.1 Gabor Filters

The Gabor filters is also known as the Gabor wavelets [5]. The method extracts features through the analysis of the frequency domain rather than the spatial domain.

The Gabor filters is represented by Equation (1) where x and y represent the pixel position in the spatial domain, ω_0 represents the radial center frequency, θ represents the orientation of the Gabor direction, and σ represents the standard deviation of the Gaussian function along the x- and y- axes where $\sigma_x = \sigma_y = \sigma$ [5].

$$\begin{aligned}\Psi(x, y, \omega_0, \theta) = \frac{1}{2\pi\sigma^2}\exp\left\{-\left((x\cos\theta + y\sin\theta)^2 + (-x\sin\theta + y\cos\theta)^2\right)/2\sigma^2\right\} \\ \times\left[\exp\{i(\omega_0 x\cos\theta + \omega_0 y\sin\theta)\} - \exp\{-\omega_0^2\sigma^2/2\}\right]\end{aligned} \quad (1)$$

The Gabor filter can be decomposed into two different equations, one to represent the real part and another to represent the imaginary part as shown in Equation (2) and Equation (3) respectively [5] while it is illustrated in Figure 1.

$$\Psi_r(x, y, \omega_0, \theta) = \frac{1}{2\pi\sigma^2}\exp\left\{-\left(\frac{x'^2 + y'^2}{\sigma^2}\right)\right\} \times \left[\cos\omega_0 x' - e^{-\omega_0^2\sigma^2/2}\right] \quad (2)$$

$$\Psi_i(x, y, \omega_0, \theta) = \frac{1}{2\pi\sigma^2}\exp\left\{-\left(\frac{x'^2 + y'^2}{\sigma^2}\right)\right\} \times \sin\omega_0 x' \quad (3)$$

where

$$x' = x\cos\theta + y\sin\theta \qquad y' = -x\sin\theta + y\cos\theta$$

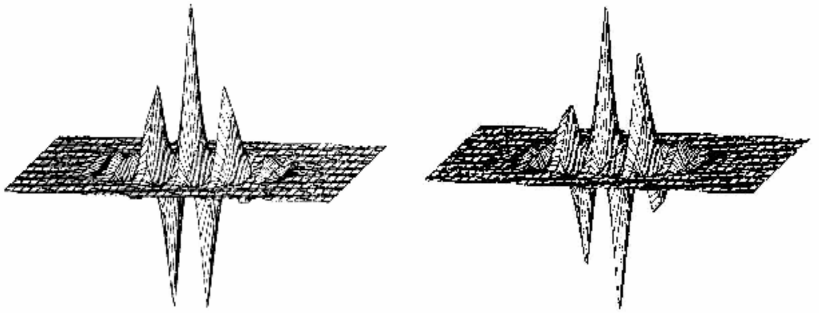


Fig. 1. Real part (left) and imaginary part (right) of a Gabor filter [8]

In this paper, we used $\sigma = \pi / \omega_0$. Gabor features are derived from the convolution of the Gabor filter Ψ and image I as shown in Equation (4) [5].

$$C_{\Psi I} = I(x, y) * \Psi(x, y, \omega_0, \theta) \tag{4}$$

The term $\Psi(x, y, \omega_0, \theta)$ of Equation (4) can be replaced by Equation (2) and Equation (3) to derive the real and imaginary parts of Equation (4) and is represented by $C_{\Psi I}^{r}$ and $C_{\Psi I}^{i}$ respectively. The real and imaginary parts are used to compute the local properties of the image using Equation (5) [5].

$$C_{\Psi I}(x, y, \omega_0, \theta) = \sqrt{\left\|C_{\Psi I}^{r}\right\|^2 + \left\|C_{\Psi I}^{i}\right\|^2} \tag{5}$$

The convolution is performed using a fast method that differs from the traditional convolution achieved through scanning windows by applying a one time convolution with Fast Fourier Transform (FFT), point-to-point multiplication and Inverse Fast Fourier Transform (IFFT). It is performed on different radial center frequencies or scales, ω_0 and orientations, θ. In this paper, the radial center frequencies and orientations are represented by θ_m in Equation (6) where $n \in \{0, 1, 2\}$ and $m \in \{0, 1, 2, \ldots, 7\}$ [5].

$$\omega_n = \frac{\pi}{2\sqrt{2}^{\,n}} \qquad\qquad \theta_m = \frac{\pi}{8} m \tag{6}$$

2.2 Covariance Matrix

A covariance matrix shows the covariance between values. In this paper, we use the fast covariance matrix calculation using integral images that is proposed in [9] to generate the covariance between different feature images to be used as the features for our algorithm.

The covariance matrix can be represented as.

$$C_R = \frac{1}{n-1} \sum_{k=1}^{n} (z_k - \mu)(z_k - \mu)^T \tag{7}$$

where z represents the feature point and μ represents the mean of the feature points for n feature points [9].

Integral images are used for faster computation. They will pre-calculate the summations for each pixel of the images from the origin point, so it is faster for the calculations of the sum for a region within the images. The calculations of the term P and Q which are two tensors for the fast calculation of the covariance matrix are shown below:

$$P(x', y', i) = \sum_{x<x', y<y'} F(x, y, i) \qquad i = 1 \ldots d \tag{8}$$

$$Q(x', y', i, j) = \sum_{x<x', y<y'} F(x, y, i) F(x, y, i) \qquad i, j = 1 \ldots d \tag{9}$$

where F represents the feature images and d represents the dimension of covariance matrix which is also the number of feature images.

The covariance matrix is then generated using P and Q where (x', y') is the upper left coordinate and (x'', y'') is the lower right coordinate of the region of interest as below [9]:

$$C_{R(x',y';x'',y'')} = \frac{1}{n-1}\Big[Q_{x'',y''} + Q_{x',y'} - Q_{x'',y'} - Q_{x'',y'} \\ -\frac{1}{n}\left(P_{x'',y''} + P_{x',y'} - P_{x',y''} - P_{x'',y'}\right)\left(P_{x'',y''} + P_{x',y'} - P_{x',y''} - P_{x'',y'}\right)^T\Big] \tag{10}$$

2.3 Nearest Neighbor

The nearest neighbor algorithm calculates the distance from the test sample against all the training samples. The best neighbor or best neighbors will be selected where a winning class is determined when it is the majority of the selected classes. In standard k-Nearest Neighbor (k-NN), the Euclidean distance is used as the metric calculation.

However, the covariance matrix does not lie on the Euclidean space, so the Euclidean distance is not suitable to be used as the metrics calculation for this case. The metrics calculation that is adopted here is using the generalized eigenvalues which is first proposed by Forstner and Moonen [10]:

$$\rho(C_1, C_2) = \sqrt{\sum_{i=1}^{n} \ln^2 \lambda_i(C_1, C_2)} \tag{11}$$

where $\lambda_i(C_1,C_2)$ represents the generalized eigenvalues of C_1 and C_2, which is computed from

$$\lambda_i C_1 x_i - C_2 x_i = 0 \qquad\qquad i = 1...d \tag{12}$$

where $x_i \neq 0$ [9].

3 Experiment Settings

In this work, the 32 textures used are from the Brodatz texture dataset [11]. This dataset were used in [7][12][13] and is shown in Figure 2. The entire Brodatz texture dataset is not used as the problem is harder to be solved for large number of classes with a limited sample size for each class respectively.

Each of the textures is separated into 16 partitions of size 64 × 64, each of the partition has four different variations, i.e. the original image, a rotated image, a scaled image and an image both rotated and scaled. Therefore there are a total of 16 sets for each texture with four samples in each set. For the training purpose, eight sets are randomly selected while the remaining eight sets are used for testing.

For the Gabor filters, a 31×31 filter was used with three radial center frequencies and eight orientations. The 24 Gabor filters are then used as the feature images to generate a 24 × 24 covariance matrix.

All experiments were tested on ten different training and testing sets which are randomly chosen using the criteria as mentioned at the first paragraph of this section.

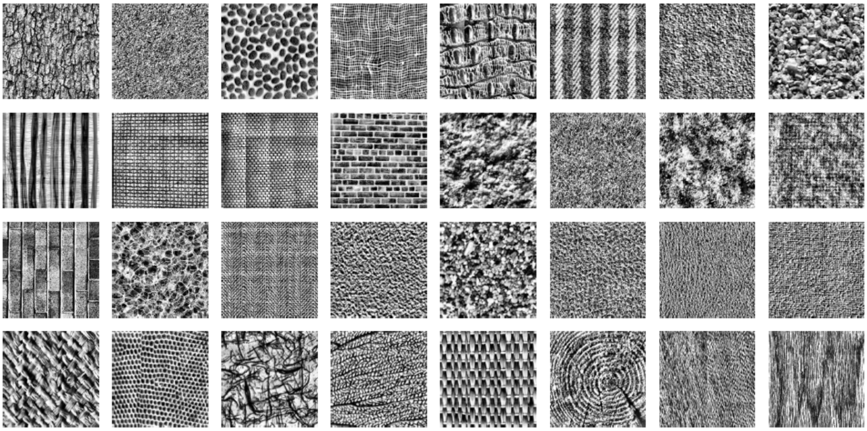


Fig. 2. 32 textures from the Brodatz texture dataset [12]

4 Results and Analysis

Three different experiments are conducted. The first experiment is done by using intensity image and its edge-based derivative images as the feature images. The second experiment is done by using four different GLCM as the feature images. The last experiment is our proposed method of using Gabor filters as the feature images.

4.1 Experiment Result for Edge-Based Derivative as Feature Images

The first experiment uses the five feature image as proposed in [9] which includes the intensity image, first derivative with respect to *x* using $[-1\ 2\ -1]^T$ filter, second derivative with respect to *x*, first derivative with respect to *y* using [-1 2 -1] filter and second derivative with respect to *y*. The feature images contain edge-based information on the vertical and horizontal directions. It generates a 5×5 covariance matrix. The accuracy achieved using this method is 84.65%.

4.2 Experiment Result for GLCM as Feature Images

The second experiment uses four GLCMs as the feature images. The GLCMs are having spatial distance of one pixel and four orientations which are 0°, 45°, 90° and 135° [13]. It generates a 4 × 4 covariance matrix. The experiment is conducted for different numbers of grey level which are 8, 16, 32, 64, 128 and 256. The results are shown in Table 1 where the horizontal bar shows the number of grey level.

Table 1. Recognition results for different numbers of grey level

(%)	8	16	32	64	128	256
Accuracy	74.56	**79.94**	75.61	69.21	56.67	41.03

The best recognition rate is 79.94% for number of grey level of 16. When the number of grey level is high, the accuracy is very much lower. At a higher number of grey levels, the variance of the GLCM within samples of the same class can vary in a greater scale due compared to those with a lower number of grey levels since similar grey values are regarded as one at lower number of grey levels, reducing the variance within samples of the same class.

4.3 Experiment Result for Gabor Filters as Feature Images

The last experiment uses the Gabor filters as the feature images for the covariance matrix. There are 3 radial center frequencies and 8 orientations used for the experiment, therefore having 24 feature images. It generates a 24 × 24 covariance matrix. Another experiment is done for only 4 orientations, therefore only generates a 12 × 12 covariance matrix. The results are shown in Table 2.

Table 2. Recognition results for Gabor filters as feature images

(%)	**12 Gabor Filters**	**24 Gabor Filters**
Accuracy	89.74	**91.86**

The best recognition rate is 91.86% for 24 Gabor filters. When fewer Gabor filters are used, the features are less and therefore the accuracy is slightly lower. The results are much better compared to the two previous experiments because the Gabor filters is able to extract frequency images in larger number that feeds in more information for the generation of covariance matrix compared to the previous techniques.

4.4 Analysis

From the results that are obtained from this paper, we compared it with the recognition rates that are achieved in our previous works as shown in Table 3.

Table 3. Comparison of recognition rates

(%)	**Accuracy**
GLCM features	85.73
Gabor features	79.87
GLCM + Gabor features	91.06
Raw GLCM	90.86
Covariance Matrix (Edge-based Derivatives)	84.65
Covariance Matrix (GLCM)	79.94
Covariance Matrix (Gabor filters)	**91.86**

From the results, we can observe that Gabor filters are not powerful enough to discriminate textures to a high accuracy when it is working by itself, but when it is combined with other techniques; it helps to improve the accuracy. For the GLCM the accuracy is higher when it works independently but when it is used to generate a covariance matrix, feature are lost as the covariance matrix is only having the size of 4 × 4 with only 10 features. Therefore, the GLCM is not suitable to be used here.

5 Conclusion

The results shows that the use of Gabor filters as the feature image for the covariance matrix is much useful than the use of edge-based derivatives or co-occurrence matrices as the feature image for the texture classification problem. The Gabor filters which are not performing as good when used independently can however produce a useful covariance matrix that can produce a better result.

Acknowledgement

This research is partly funded by Malaysian MOSTI ScienceFund 01-02-11-SF0019.

References

1. Tuceryan, M., Jain, A.K.: Texture analysis. In: The Handbook of Pattern Recognition and Computer Vision, 2nd edn., pp. 207–248. World Scientific Publishing, Singapore (1998)
2. Lew, Y.L.: Design of an Intelligent Wood Recognition System for the Classification of Tropical Wood Species. M.E. Thesis, Universiti Teknologi Malaysia (2005)
3. Tou, J.Y., Lau, P.Y., Tay, Y.H.: Computer Vision-based Wood Recognition System. In: Proc. Int'l Workshop on Advanced Image Technology, Bangkok (2007)
4. Partio, M., Cramariuc, B., Gabbouj, M., Visa, A.: Rock Texture Retrieval using Gray Level Co-occurrence Matrix. In: Proc. of 5th Nordic Signal Processing Symposium (2002)
5. Yap, W.H., Khalid, M., Yusof, R.: Face Verification with Gabor Representation and Support Vector Machines. In: IEEE Proc. of the First Asia International Conference on Modeling and Simulation (2007)
6. Chen, Y.Q.: Novel techniques for image texture classification. PhD Thesis, University of Southampton, United Kingdom (1995)
7. Tou, J.Y., Tay, Y.H., Lau, P.Y.: Gabor Filters and Grey-level Co-occurrence Matrices in Texture Classification. In: MMU International Symposium on Information and Communications Technologies, Petaling Jaya (2007)
8. Nixon, M., Aguando, A.: Feature extraction and Image Processing. Butterworth-Heinemann, Great Britain (2002)
9. Tuzel, O., Porikli, F., Meer, P.: Region Covariance: A Fast Descriptor for Detection and Classification. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 697–704. Springer, Heidelberg (2006)
10. Forstner, W., Moonen, B.: A Metric for Covariance Matrices. Technical report, Dept. of Geodesy and Geoinformatics, Stuttgart University (1999)
11. Brodatz, P.: Textures: A Photographic Album for Artists and Designers. Dover, New York (1996)
12. Ojala, T., Pietikainen, M., Kyllonen, J.: Gray level Cooccurrence Histograms via Learning Vector Quantization. In: Proc. 11th Scandinavian Conference on Image Analysis, pp. 103–108 (1999)
13. Tou, J.Y., Tay, Y.H., Lau, P.Y.: One-dimensional Grey-level Co-occurrence Matrices for Texture Classification. In: International Symposium on Information Technology, Kuala Lumpur, vol. 3, pp. 1592–1597 (2008)

Investigating Demographic Influences for HIV Classification Using Bayesian Autoassociative Neural Networks

Jaisheel Mistry[1], Fulufhelo V. Nelwamondo[2], and Tshilidzi Marwala[1]

[1] University of Witwatersrand, School of Electrical and Information Engineering,
1 Jans Smuts, 2050 Braamfontein, South Africa
Jaisheel.Mistry@wits.ac.za,, Tshilidzi.Marwala@wits.ac.za

[2] Harvard University, Graduate School of Arts Sciences, Cambridge,
Massachusetts,USA
nelwamon@fas.harvard.edu

Abstract. This paper presents a method of determining whether demographic properties such as education, race, age, physical location, gravidity and parity influence the ability to classify the HIV status of a patient. The degree to which these variables influence the HIV classification is investigated by using an ensemble of autoassociative neural networks that are trained using the Bayesian framework. The HIV classification is treated as a missing data problem and the ensemble of autoassociative neural networks coupled with an optimization technique are used to determine a set of possible estimates. The set of possible estimates are aggregated together to give a predictive certainty measure. This measure is the percentage of the most likely estimate from all possible estimates. Changes to the state of each of the demographic properties are made and changes in the predictive certainty are recorded. It was found that the education level and the race of the patients are influential on the predictability of the HIV status. Significant knowledge discovery about the demographic influences on predicting a patients HIV status is obtained by the methods presented in this paper.

1 Introduction

Neural networks and evolutionary programming have successfully been used to model various nonlinear problems in the field of medical informatics. Computational intelligence and artificial intelligence have been used successfully for decision making, clinical diagnosis, prognosis and prediction of outcomes. One such problem in this field is for better understanding the HIV/AIDS pandemic. Some of the research on this problem include investigating the causes the HIV AIDS virus [1,2], predicting the HIV status for risk analysis purposes [3,4] and to better understand the risks of such a virus [5]. In the field of bioinformatics HIV classifications using neural networks are presented in [5,6,7,8].

In this paper the demographic influences on a patients HIV status are investigated by using the computational intelligence missing data estimation method.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 752–759, 2009.

The predictive certainty of the HIV status is found by using an ensemble of autoassociative neural networks. Using the ensemble of classifiers for obtaining a set of possible solutions allows for obtaining predictive certainty measures as suggested in [9]. The Bayesian approach will be used for training to obtain an ensemble of neural networks. The estimate from each autoassociative neural networks is aggregated using a voting scheme and the predictive certainty is measured by giving the percentage of the most likely estimate. The change in predictive certainty is measured for adjustments made to the possible states of the different demographic properties of the patients. The changes in predictive certainty will help one better understand the demographic influences on classifying a patients HIV status. A background on Bayesian autoassociative neural networks and the computational intelligence method for missing data estimation is presented. The method of investigation is then given. The results are discussed and conclusions are made.

2 Background

2.1 Bayesian Autoassociative Neural Networks

Autoassociative neural networks are neural networks that have fewer hidden nodes than input nodes [10]. This implies that the inputs to the network are projected to a smaller dimension. This dimensionality reduction is particularly applicable to data that has been encoded using a unary coding scheme.Hence the feed foward network architecture autoassociative neural network is guaranteed of removing the redundancy of using such coding schemes [10]. The autoassociative bottle-neck structure can be thought of as reducing dimensionality of the input data by taking account of the covariance and correlation of the various dimensions of data [11].

Neural Networks can be trained using a variety of methods. Scaled Conjugate Gradient Methods and Quasi-Newton Methods are used to obtain the maximum likelihood weight values. An alternate approach to training the neural networks is using the Bayesian Approach. Extensive work has been done by [12,13] to train neural networks using the Bayesian approach. The problem of identifying the weights (w) is posed in Bayesian form as follows [12]:

$$P(w|D) = \frac{P(D|w)P(w)}{P(D)} \tag{1}$$

where P(w) is the probability distribution function of the weight-space in the absence of any data, also known as the prior probability distribution function, and $D \equiv (y1, , yN)$ is a matrix containing the output data for the neural network. The quantity $P(w|D)$ is the posterior probability distribution after the data have been seen and $P(D|w)$ is the likelihood probability distribution function, while $P(D)$ is the normalization factor. Following the rules of probability theory, the distribution of output vector y may be written in the following form:

$$P(y|D) = \int P(y|w)P(w|D)dw \tag{2}$$

The distribution in equation 2 is estimated using Markov Chain Monte Carlo (MCMC) method. In this paper, the Hybrid Monte Carlo (HMC) MCMC method is used.

2.2 Computational Intelligence Missing Data Estimation Method

Many methods are used for estimating missing data which include zero substitution, mean substitution, look up tables and statistical regression. Other missing data estimation methods are found in [14,15]and [16].The Computational Intelligence method for estimating missing data is initially presented in [17]. A review of this method is given in [18]. The general method to estimate missing data consists of an autoassociative neural network (or any other autoassociative model such as decision tree) coupled with an optimization technique . Suppose known data are denoted by X_K and data that is unknown or missing are denoted by X_U. The optimization technique is used to minimize an error function of the estimated unknown value $X_U^{'}$ and the corresponding output $F\left\{X_U^{'}\right\}$ from the autoassociative neural network. The autoassociative neural network $F\left\{\right\}$ is a system that recognizes patterns or interrelationships of variables in the data. The flow diagram of this computational intelligence missing data estimation method is shown in figure 1.This identification system is obtained by training on a set of complete data so that the dynamics of the system are identified. When a minimum error is calculated the corresponding $X_U^{'}$ is stored as the estimate value. The error of the predicted output of the identification system is calculated as follows:

$$error = \left(\begin{bmatrix} X_K \\ X_U \end{bmatrix} - F\left\{\begin{bmatrix} X_K \\ X_U \end{bmatrix}\right\}\right)^2 \tag{3}$$

where the matrix $\begin{bmatrix} X_K \\ X_U \end{bmatrix}$ is representative of a single record of known and unknown data. Optimization techniques such as Genetic Algorithm, Particle Swarm Optimization or any other stochastic optimization techniques can be used to find $X_U^{'}$ that minimizes the error in equation 3. This estimation method is not dependent on how many variables are missing or which variables are missing. This

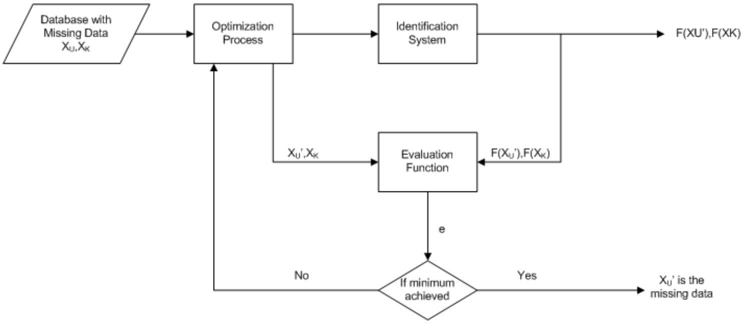


Fig. 1. Flow Diagram of Computational Intelligence Missing Data Estimation Method

method is found to work more accurately than using neural network classifiers because errors are computed for a higher dimension. If the search space for X_U is small (for example when only 1 variable is missing), a brute force approach to evaluating the error for all possible estimations can also be used.

3 Method

The data used for the investigation as well as the training of the neural networks are further discussed. The optimization method for determining the missing data is also discussed.

3.1 HIV Dataset

The dataset used to carry out this investigation is that of demographic properties from the South African antenatal seroprevalence survey. The different fields of data for each record includes age of patient, age of partner, provincial location, race group, maximum education level achieved, gravidity, parity, Rapid Plasma Reagin(RPR) testing result and HIV status. A simple unary coding scheme was used to create binary inputs for the fields such as province, race, education, HIV status and RPR test. Hence 4 binary inputs were used to represent maximum education level, 9 binary inputs are used to represent province, 5 binary inputs are used to represent race group and a single binary bit is used to represent HIV status and another bit is used to indicate whether a patient has taken a RPR test.

3.2 Network Training

The Multi Layer Perceptron (MLP) architecture is used for the autoassociative neural network investigated in this paper. Considering the unary coding scheme used, the autoassociative neural network has 23 inputs. As explained in section 2.1 the HMC training method gives a set of possible weights for certain number of hidden nodes. The weights were initially trained using the scaled conjugate gradient method to the early stopping point. The benefits of setting the weights to this prior value are discussed in [19]. The burn in period was set to 300 cycles and samples were drawn from the following 200 cycles. Duplicate samples were removed so that the ensemble contained only unique samples. The ensemble of autoassociative neural networks was obtained by collecting a set of possible weights for MLP neural networks trained with 15 to 23 hidden nodes in order to increase the structural diversity [9].

3.3 Missing Data Estimation

For purposes of investigation in this paper the HIV status of each record in the test set will be assumed missing and the method described in section 2.2 is used to estimate the HIV status of each patient. A similar investigation to classify the HIV status using computational intelligence method was done in

[3] and [4]. This investigation differs in that it is done using the ensemble of trained autoassociative neural networks. There is no need to use a complicated optimization technique because the possible state of the missing variable is either 1 for HIV positive or 0 for HIV negative. Hence the error function in equation (3) is evaluated for both possible states and the state that yields the minimum error is used as the optimum estimate.

This brute force approach for obtaining the optimum state to minimize the error is used for each of the autoassociative neural networks from the ensemble. The estimates are aggregated together by selecting the most popular estimate. The predictive certainty measure of the estimate is given by calculating the percentage of the most dominant estimate from the set of possible estimates.

3.4 Influences for HIV Classification

The demographic influences of the HIV virus are investigated by means of a sensitivity analysis. In sensitivity analysis we compare the model output with the produced output for the modified input parameters. The sensitivity analysis done in this chapter measures the changes the change in predictive certainty with a change of state of the various demographic influences. In this paper the sensitivity analysis is done for records that can be predicted with a high predictive certainty. The changes in predictive certainty obtained when changing the variables helps understand the impact each variable has on classifying a patients HIV status.

4 Results and Discussion

A set of 120 autoassociative neural networks were obtained using the HMC training. The predictive capabilities of these networks were tested using a validation dataset. It was found that the accuracy of these networks range between 77% and 96%.

The ensemble of neural networks was used together with the brute force optimization algorithm to yield an overall accuracy of 68% for the missing data estimation. This overall prediction accuracy exceeds the accuracy achieved in [3]. It was found that only 40% of the HIV status could be estimated with a predictive certainty greater than 70%. For the records that could be estimated with a predictive certainty of 70% it was found that 88% of these records were correctly estimated. The method disscussed in section 3.4 is used for determining how the patients maximum education achieved, race group, age of husband, gravidity, parity and provincial location influenced the HIV classification. These results are presented in tables 1 to 6.

From table 1 to 6 it is evident that there is a bigger change in predictive certainty for patients that are classified as HIV positive. This is because there are fewer cases where patients are classified as HIV positive. A change in predictive certainty of greater than 40% would result in the patients HIV status changing. These large changes in predictive certainty are evident in the tables 1, 2 and 7. Therefore there is allot of uncertainty for predicting a patients HIV status when

Table 1. Average change in Predictive Certainty(%) for HIV positive and HIV negative patients for changes in maximum education level achieved

HIV status	uneducated	primary	seondary	tertiary
Negative	39	11	3	15
Positive	18	12	0	42

Table 2. Average change in Predictive Certainty(%) for HIV positive and HIV negative patients for changes in Race Type(RT)

HIV status	RT1	RT2	RT3	RT4
Negative	33	2	9	24
Positive	2	41	58	51

Table 3. Average change in Predictive Certainty(%) for HIV positive and HIV negative patients for changes in patients partner age group(GP)

HIV status	GP1	GP2	GP3	GP4	GP5	GP6	GP7	GP8
Negative	2	1	2	2	4	7	11	15
Positive	4	2	2	5	11	17	23	27

Table 4. Average change in Predictive Certainty(%) for HIV positive and HIV negative patients for changes in gravidity ranging from 0 to 5

HIV status	0	1	2	3	4	5
Negative	11	6	3	2	2	5
Positive	9	1	9	20	25	31

Table 5. Average change in Predictive Certainty(%) for HIV positive and HIV negative patients for changes in parity ranging from 0 to 5

HIV status	0	1	2	3	4	5
Negative	6	3	3	6	10	12
Positive	1	10	20	25	21	34

Table 6. Average change in Predictive Certainty(%) for HIV positive and HIV negative patients for changes in Province(P)

HIV status	P1	P2	P3	P4	P5	P6	P7	P8	P9
Negative	19	47	51	14	24	2	20	29	23
Positive	2	2	15	15	3	33	24	28	31

using education level, race group and provincial location. Given the high degree of uncertainty when using these variables and their particular state, one should be less confident at classifying such a patients HIV status.

This research was aimed at better understanding the demographic influences for HIV classifications. Although all the demographic variables influence the ability to classify a patients HIV status, the research shows that certain variables such as race group, educational level and provincial location influence the ability to classify a patients HIV status with high degree of uncertainty. Similar research was done in [1], where inverse neural networks were used for adaptive control of the HIV virus. In [1] gravidity and educational level were assumed influential and used to build inverse neural networks that helped determine how they affect the risk of being HIV positive. The research in this paper verfies this and also shows that some of the other variables are influential to such an extent that there is a high uncertainty in predicting a patients HIV status.

5 Conclusion

The demographic influences for classifying patients HIV status was investigated using using Bayesian autoassociative neural networks with computational intelligence missing data estimation methods. The high variability in predictive certainty shows that the HIV system being investigated performs classification with a high degree of uncertainty. It is evident that changing the demographic properties of HIV positive patients has more variability in predictive certainty. It was found that education elevel,race group and provincial location are the most influential for the HIV status classification and the age of mother and patient of partner have minimal influence on classifying a patients HIV status. The methods applied in this paper help with knowledge discovery regarding the demographic influences for HIV classification.

References

1. Marwala, T., Leke, B.B., Tettey, T.: Using inverse neural network for HIV adaptive control. International Journal of Computational Intelligence Research 3, 11–15 (2007)
2. Lee, C.W., Park, J.-A.: Assessment of HIV/AIDS-related health performance using an artificial neural network. Information and Management 38(4), 231–238 (2001)
3. Leke, B.B., Abdella, M., Tim, T., Lagazio, M.: Autoencoder networks for HIV classification. Current Science 91, 1467–1473 (2006)
4. Leke, B.B., Marwala, T., Tim, T., Lagazio, M.: Prediction of HIV status from demographic data using neural networks. In: 2006 IEEE International Conference on Systems, Man and Cybernetics, Taipei, Taiwan, vol. 3, pp. 2339–2344 (2007)
5. Leke, B.B., Tim, T., Marwala, T., Lagazio, M.: Using genetic algorithms versus line search optimization for HIV predictions. WSEAS Transactions on Information Science and Applications 3(4), 684–690 (2006)
6. Berger, E.A., Doms, R.W., Fenyo, E.M., Korber, B.T.M., Littman, D.R., Moore, J.P., Sattentau, Q.J., Schuitemaker, H., Sodroski, J., Weiss, R.A.: New classification for HIV-1. Nature 391(6664), 240 (1998)

7. Srisawat, A., Kijsirikul, B.: Using associative classification for predicting HIV-1 drug resistance. In: 4th International Conference on Hybrid Intelligent Systems, Kitakyushu, Japan, December 5-8, 2004 (2005)
8. Wang, M., Zheng, J., Chen, Z., Shi, Y.: Classification methods for HIV-1 medicated neuronal damage. In: 2005 IEEE Computational Systems Bioinformatics Conference, Stanford, CA, United States, pp. 31–32 (2005)
9. Polikar, R.: Ensemble based systems in decision making. IEEE Circuits and Systems Magazine 6(3), 21–44 (2006)
10. Kramer, M.A.: Autoassociative neural networks. Computers & Chemical Engineering 16(4), 313–328 (1992)
11. Mistry, J., Nelwamondo, F.V., Marwala, T.: Using principal component analysis and autoassociative neural networks to estimate missing data in a database. In: Proceedings of World Multi Conference on Systemics, Cybernetics and Informatics, July 2008, vol. 4, pp. 24–29 (2008)
12. Neal, R.M.: Probabilistic inference using markov chain monte carlo methods. Computer science tech report crg-tr-93-1, University of Toronto (1993)
13. MacKay, D.J.C.: Bayesian methods for neural networks: Theory and applications. Course notes for Neural Networks Summer School (1995)
14. Rubin, D.B.: Multiple imputation for nonresponse in surveys, pp. 20–34 (1978)
15. Little, R.J.A., Rubin, D.B.: Statistical analysis with missing data. Wiley, Hoboken (2002)
16. Allison, P.D.: Multiple imputation for missing data: A cautionary tale. Sociological Methods and Research 28(3), 301–309 (2000)
17. Abdella, M., Marwala, T.: The use of genetic algorithms and neural networks to approximate missing data in database. IEEE 3rd International Conference on Computational Cybernetics 24, 207–212 (2005)
18. Nelwamondo, F.V., Marwala, T.: Key issues on computational intelligence techniques for missing data imputation - a review. In: Proceedings of World Multi Conference on Systemics, Cybernetics and Informatics, July 2008, vol. 4, pp. 35–36–40 (2008)
19. Vehtari, A., Sarkka, S., Lampinen, J.: On MCMC sampling in bayesian mlp neural networks. In: International Joint Conference on Neural Networks (IJCNN 2000), vol. 1, pp. 317–322. Helsinki Univ of Technology, Finl, Como, Italy (2000)

Hardware-Based Solutions Utilizing Random Forests for Object Recognition

Hassab Elgawi Osman

School of Engineering, Tokyo Institute of Technology, Japan
osman@isl.titech.ac.jp

Abstract. This paper presents how hardware-based machine learning models can be designed for the task of object recognition. The process is composed of automatic representation of objects as covariance matrices follow by a machine learning detector based on random forest (RF) that operate in on-line mode. We describe the architecture of our random forest (RF) classifier employing Logarithmic Number Systems (LNS), which is optimized towards a System-on-Chip (Soc) platform implementation. Results demonstrate that the proposed model yields in object recognition performance comparable to the benchmark standard RF, AdaBoost, and SVM classifiers, while allow fair comparisons between the precision requirements in LNS and of using traditional floating-point.

1 Introduction

Random forests (RFs) [8], a representative decision tree-based ensemble learning technique has been emerging as a principle machine learning tool combining properties of an efficient classifier and feature selection. In addition to its popularity in classification and regression applications, RF also has been applied to object recognition [3,6] but only for a relatively small number of classes. Despite of the appearance success of RF little work has been done exploring this technique into on-line settings, virtually no work has been done to map from its ideal mathematical model to compact and reliable hardware design. In this paper we extend this technique into on-line mode [4,5], and then present our object recognition system which is optimized to be easily integrated in a System-on-Chip (SoC). As can be seen in Fig. 1 the recognition process is composed of automatic representation of objects as covariance matrices follow by random forest (RF) detector that operate in on-line mode. A random forest detector is designed using Logarithmic Number Systems (LNS) yielding in a considerable hardware savings with no significant degradations in recognition accuracy. The architecture comprises several computation modules, referred to as ‘Covariance Matrices’, ‘Tree Units’, ‘Majority Vote Unit’, and ‘Forest Units’. However, it should be noted that the number of these Units that can be accommodated depends on the hardware platform. We present results obtained using examples from GRAZ02 dataset [2] and compares with state of art machine learning classifiers.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 760–767, 2009.

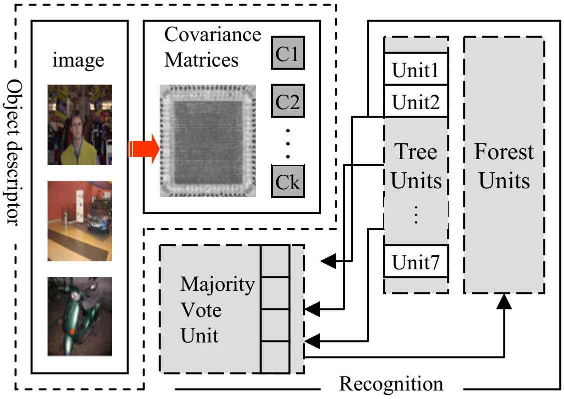


Fig. 1. Object Recognition based on RF-LNS which is optimized to be easily integrated in a System-on-Chip (SoC) platform implementation

2 Proposed Object Descriptor Method

We have used bag of covariance[1] matrices, to represent an object region. Let I be an input color image. Let F be the dimensional feature image extracted from I

$$F(x, y) = \phi(I, x, y) \tag{1}$$

where the function ϕ can be any feature maps (such as intensity, color, etc). For a given region $R \subset F$, let $\{f_j\}_{j=1\cdots n}$ be the d dimensional feature points inside R. We represent the region R with the $d \times d$ covariance matrix C_R of feature points.

$$C_R = \frac{1}{n-1} \sum_{j=1}^{n} (f_j - \mu)(f_j - \mu)^T \tag{2}$$

where μ is the mean of the point. Fig.1 (i) depicts the points that must be sampled around a particular point (x, y) in order to calculate the histograms of Local Binary Patterns (LBP) at (x, y). In our implementation, each sample point lies at a distance of 2 pixels from (x, y), instead of the traditional 3×3 rectangular neighborhood, we sample neighborhood circularly with two different radii (1 and 3). The resulting operators are denoted by $LBP_{8,1}$ and $LBP_{8,1+8,3}$, where subscripts tell the number of samples and the neighborhood radii. In Fig.1 (ii), different regions of an object may have different descriptive power and hence, difference impact on the learning and recognition.

2.1 Labeling the Image

We gradually build our knowledge of the image, from features to covariance matrix to a bag of covariance matrices. Our first step is to model each covariance matrix as a set of

[1] Basically, covariance is a measure of how much two variables vary together.

image features. Next, we group covariance matrices that are likely to share common label into a bag of covariance matrices. We follow [9] and represent an image objects with five covariance matrices $C_{i=1\cdots5}$ of the feature computed inside the object region, as shown in the second row of Fig.1. A bag of covariance which is necessary a combination of Ohta color space histogram ($I_1 = R+G+B/3$, $I_2 = R-B$, $I_3 = (2G-R-B)/2$), LBP and appearance model of different features of an image window is presented in Fig.1 (iii). Then estimate the bag of covariance matrix likelihoods and the likelihood that each bag of covariance matrices is homogeneously labeled. We use this representation to automatically detect any target in images. We then apply on-line RF learner to select object descriptors and to learn an object classifier, as can be seen in the last row of Fig.2.

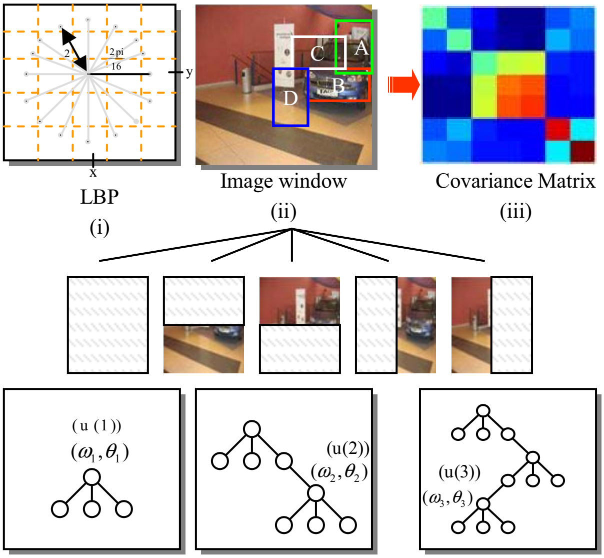


Fig. 2. (i) Points sampled to calculate the LBP around a point (x, y). (ii) Rectangles are examples of possible regions for histogram features. Stable appearance in Rectangles A, B and C are good candidates for a car classifier while regions D is not. (iii) Any region can be represented by a covariance matrix. Size of the covariance matrix is proportional to the number of features used. Second row shows an object represented with five covariance matrices.

3 Hardware Architecture

Details discussion of Breiman's random forest (RF) [8] learning algorithm is beyond the scope of this paper. We refer the reader to [4,5] for a good introduction to the on-line setting of random forest and the details of solutions to classification and vision.

3.1 Logarithmic Number Systems (LNS)

LNS are alternative to fixed-and floating-point arithmetic. Within logarithm domain, multiplication and division can be treated simply as addition or subtraction, the number of bits and the frequency of their switching are significantly reduces. Hardware computation of these operations is significantly faster with reduced complexity. Unlike Floating-Point (FP) systems, the relative error of LNS is constant and LNS can often achieve equivalent signal-to-noise ratio with fewer bits of precision relative to conventional FP architectures. Similar to FP architectures, LNS implementations can represent numbers with relative precision; numbers closer to zero, are represented with better precision in LNS than FP systems. LNS are particularly cost effective when an application performs acceptably with reduced precision.

3.2 RF-LNS

The Covariance Unit in Fig. 1 contains all the features extracted from an image in a form of bag of covariance matrices. To obtain RF-LNS, each base learner (decision trees) is treated as a Tree Unit, estimated by a single covariance matrix selected from bag of covariance. Each least node in the tree is associated to a given object class. Basically the decision trees consist of two types of nodes: *decision nodes* and *least nodes*, which correspond to all possible covariance features that can be taken. In a decision node a decision is taken about one of the input. Each least node stores the values for the corresponding region in the image, meaning that a least node stores a value for each relevant covariance matrix that can be taken. The tree starts out with only one leaf that represents the entire image region. So in a least node a decision has to be made whether the node should be split or not. Once a tree is constructed it can be used to map an input vector to a least node, which corresponds to a region in the image. Each tree gives a unit vote for its popular object class. The object is recognized as the one having the majority vote. Forest Unit is ensemble of trees grown incrementally to a certain depth.

3.3 RF-LNS Implementation

In regard to implementation, the final decision function requires at least on multiplication and one addition for each decision tree. Within the logarithmic domain, multiplication and division can be treated simply as addition or subtraction. Hardware computation of these operations is significantly fast with reduced complexity.

4 Object Recognition

Given a feature set and a sample set of positive (contains the object relevant to the class) and negative (does not contain the object) images, to detect a specific object, e.g. human, in a given image, the main difficulty is to train a classifier with relevant features toward accurate object recognition. The adoption of RF learner and its ability to measure feature importance relief us from this challenge. We train a random forests learner (detector) offline using covariance descriptors of positive and negative samples. We start by evaluation feature from input image I after the detector is scanned over it at multiple locations and scales. This has to be done for each object. Then for feature in I, we want to find corresponding covariance matrix for estimating a decision tree. Each decision tree learner may explore any feature f, we keep continuously accepting or rejecting potential covariance matrices. We then apply the on-line random forests at each candidate image window to determine whether the window depicts the target object or not. The on-line RF detector was defined as a 2 stage problem, with 2 possible outputs: In the first stage, we build a detector that can decide if the image contains an object, and thus must be recognized, or if the image does not contain objects, and can be discarded, saving processing time. In the second stage, based on selected features the detector must decide which object descriptor should be used. There are two parameters controlling the learning recognition process: The depth of the tree, and the least nodes. It is not clear how to select the depth of the on-line forests. One alternative is to create a growing on-line forests where we first start with an on-line forest of depth one. Once it converges to a local optimum, we increase the depth. Thus, we create our on-line forest by iteratively increasing its depth.

4.1 Detection Instances

Next, when detecting a new instance, we first estimate the average margin of the trees on the instances most similar to the new instance and then, after discarding the trees with negative margin, weight the tree's votes with the margin. Then the set of classifiers is updated. For updating, any on-line learning algorithm may be used, but we employ a standard Karman filtering technique [7] and build our updated model by estimate the probability $P(1|f_jx)$ with mean μ^+ and standard deviation σ^+ for positive samples and $P(-1|f_j(x))$ by $N(\mu^-, \sigma^-)$ for negative samples similar way as we do in the off-line case.

4.2 Dataset

We now demonstrate the usefulness of this frame work in the area of recognition generic objects such as bikes, cars, and persons. We used data derived from the GRAZ02[2] dataset [2], a collection of 640×480 24-bit color images. As Fig.3 illustrates, the GRAZ02 database contains variability with respect to scale and clutter. Objects of interest are often occluded, and they are not dominant in the image. According to [1] the average ratio of object size to image size counted in number of pixels is 0.22 for bikes, 0.17 for people, and 0.9 for cars. Thus this dataset is more complex dataset to learn detectors from, but of more interest because it better reflects the real world complexity. This dataset has three object classes, bikes (365 images), cars (420 images) and persons (311 images), and a background class (270 images).

4.3 Experimental Settings

Our RF-LNS is trained with varying amounts (10%, 50% and 90% respectively) of randomly selected training data. All image not selected for the training split were put into the test split. For the 10% training data experiments, 10% of the image were selected randomly with the remainder used for testing. This was repeated 20 times. For the 50% training data experiments, stratified 5×2 fold cross validation was used. Each cross

[2] Available at *http://www.emt.tugraz.at/∼pinz/data/*

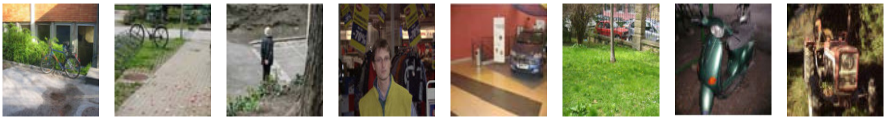


Fig. 3. Examples from GRAZ02 dataset [2] for four different categories: bikes (1st pair), people (2nd pair), cars (3rd pair), and background (4th pair)

validation selected 50% of the dataset for training and tested the classifiers on the remaining 50%; the test and training sets were then exchanged and the classifiers retrained and retested. This process was repeated 5 times. Finally, for the 90% training data situation, stratified 1×10 fold cross validation was performed, with the dataset divided into ten randomly selected, equally sized subsets, with each subset being used in turn for testing after the classifiers were trained on the remaining nine subsets.

5 Performances

GRAZ02 images contain only one object category per image so the recognition task can be seen as a binary classification problem: bikes vs. background, people vs. background, and car vs. background. Generalization performances in these object recognition experiments were estimated by statistic measure; the Area Under the ROC Curve (AUC) to measure the classifiers performance. The AUC is a measure of classifier performance that is independent of the threshold: it summarizes not the accuracy, but how the true positive and false positive rate change as the threshold gradually increases from 0.0 to 1.0. An ideal, perfect, classifier has an AUC value 1.0 while a random classifier has an AUC of 0.5.

5.1 Finite Precision Analysis

The primary task here is to analyze the precision requirements for performing on-line RF classification in LNS hardware. The LNS precision in the RF algorithm was varied to ascertain optimal LNS precision and compare them against the cost of using traditional floating-point architectures. Tables 1 gives the mean AUC values across all runs to 2 decimal places for RF-LNS and training data amount combinations, for the bikes, cars ad people datasets. The performance of RF-LNS is reported with weight quantized with 4, 8, and 16 bits, and for different depths of the tree from depth = 3 to depth = 7. In order to maintain acceptable performance, 16 bits of precision are sufficient for all the datasets. In order to evaluate the efficiency of an LNS-based RF classifier, it is necessary to compare it against a traditional standard. To that extent, 10- and 20-bit Fixed Point (FX) implementations were synthesized, and the resulting numbers of slices are shown in Table 2. It is note worthy that on most datasets; the fully functional LNS version takes roughly the same number of slices as the inadequate 10-bit FX version. When compared against the more realistic 20-bit FX version, the LNS classifiers are about one-half the size of the FX classifiers. Such area savings should translate into equivalent reduction in power consumption.

Table 1. Mean AUC performance of RF-LNS on the Bikes vs. Background, the Cars vs. Background, and Persons vs. Background dataset, by amount of training data. Performance of RF-LNS is reported for different Depths, with weight quantized with 4, 8, and 16 bits.

	Bikes vs. Background									
	RF-LNS (4-bit Precision)					RF-LNS (8-bit Precision)				
	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7
10%	0.79	0.79	0.77	0.81	0.81	0.81	0.81	0.80	0.83	0.83
50%	0.86	0.86	0.82	0.81	0.83	0.88	0.89	0.85	0.88	0.86
90%	0.80	0.81	0.81	0.83	0.88	0.87	0.87	0.87	0.88	0.90
	RF-LNS (16-bit Precision)									
	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7					
10%	0.83	0.83	0.81	0.84	0.83					
50%	0.90	0.90	0.86	0.89	0.89					
90%	0.90	0.91	0.90	0.90	0.90					
	Cars vs. Background									
	RF-LNS (4-bit Precision)					RF-LNS (8-bit Precision)				
	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7
10%	0.66	0.70	0.70	0.75	0.71	0.68	0.73	0.73	0.76	0.73
50%	0.77	0.78	0.77	0.77	0.79	0.79	0.80	0.79	0.81	0.81
90%	0.77	0.75	0.75	0.73	0.79	0.81	0.81	0.78	0.78	0.82
	RF-LNS (16-bit Precision)									
	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7					
10%	0.71	0.75	0.75	0.77	0.75					
50%	0.81	0.80	0.81	0.82	0.83					
90%	0.83	0.83	0.81	0.80	0.85					
	Persons vs. Background									
	RF-LNS (4-bit Precision)					RF-LNS (8-bit Precision)				
	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7
10%	0.66	0.70	0.70	0.75	0.71	0.68	0.73	0.73	0.76	0.73
50%	0.77	0.78	0.77	0.77	0.79	0.79	0.80	0.79	0.81	0.81
90%	0.77	0.75	0.75	0.73	0.79	0.81	0.81	0.78	0.78	0.82
	RF-LNS (16-bit Precision)									
	Dth=3	Dth=4	Dth=5	Dth=6	Dth=7					
10%	0.71	0.75	0.75	0.77	0.75					
50%	0.81	0.80	0.81	0.82	0.83					
90%	0.83	0.83	0.81	0.80	0.85					

Table 2. Slices used for different Tree Units for each dataset during

Bikes				Cars				Persons			
Tree	16bit	10FX	20FX	Tree	16bit	10FX	20FX	Tree	16bit	10FX	20FX
3	315	219	576	3	277	283	603	3	336	318	409
4	498	407	713	4	297	476	783	4	534	535	657
5	611	622	878	5	536	694	866	5	765	689	845
6	823	835	1103	6	784	943	1002	6	878	926	1127
7	1010	974	1345	7	989	1287	1311	7	1123	1158	1287

6 Conclusions and Future Works

Efficient hardware implementations of machine-learning techniques yield a variety of advantages over software solutions: increased processing speed, and reliability as well as reduced cost and complexity. This paper describes preliminary research towards the development of robust, friendly hardware-based solutions utilizing RF for Object recognition task. Our hardware-friendly algorithm that has been described here is general towards the implementation of on-chip learning, which is efficient in terms of hardware complexity. Our future goals are expanding LNS hardware architectures to other machine-learning algorithms.

References

1. Opelt, A., Pinz, A.: Object Localization with Boosting and Weak Supervision for Generic Object Recognition. In: Kalviainen, H., Parkkinen, J., Kaarna, A. (eds.) SCIA 2005. LNCS, vol. 3540, pp. 862–871. Springer, Heidelberg (2005)
2. Oplet, A., Fussenegger, M., Pinz, A., Auer, P.: Generic object recognition with boosting. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(3), 416–431 (2006)
3. Moomsmann, F., Triggs, B., Jurie, F.: Fast discriminative visual codebooks using randomized clustering forests. In: NIPS 2006 (2006)
4. Hassab Elgawi, O.: Online Random Forests based on CorrFS and CorrBE. In: Proc. IEEE workshop on online classification, CVPR, pp. 1–7 (2008)
5. Hassab Elgawi, O.: Incremental Machine Learning Approach for Component-based Recognition. In: Proc. Int'l. Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2009) (2009)
6. Winn, J., Criminisi, A.: Object class recognition at a glance. In: CVPR (2006)
7. Karman, K.P., von Brandt, A.: Moving object recognition using an adaptive background memory. In: Capellini (ed.) Time-varying Image Processing and Moving Object Recognition, vol. II, pp. 297–307. Elsevier, Amsterdam (1990)
8. Breiman, L.: Random Forests. Machine Learning 45(1), 5–32 (2001)
9. Tuzel, O., Porikli, F., Meer, P.: Region covariance: A fast descriptor for detection and classification. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 589–600. Springer, Heidelberg (2006)

A Neural Oscillation Model for Contour Separation in Color Images [*]

Yu Ma, Xiaodong Gu, and Yuanyuan Wang[**]

Department of Electronic Engineering, Fudan University, Shanghai, 200433, China
{mayu,xdgu,yywang}@fudan.edu.cn

Abstract. A novel neural oscillation model is proposed to perform the contour detection and separation for color images. The model improves the prototype of relaxation oscillation and combines four kinds of image features as the external stimulation, in order to control the oscillation. Experimental results show that the contours of different objects can be detected by oscillation and separated by desynchronization. This model may help to promote the investigation on the mechanism of the neural system.

1 Introduction

The visual system has the ability of grouping different kinds of features, which underlie a variety of tasks such as image segmentation, contour detection and object recognition. The principle of this ability has been assumed differently and many different theories are built to describe it. In the 1980s, Von der Malsburg proposed the temporal correlation theory to explain the feature grouping. The theory considered that an object is represented by the temporal correlation of the firing activities of scattered cells which code different features [1]. Later in 1989, the coherent oscillation phenomenon was discovered in the visual cortex of the cat [2, 3]. This discovery not only strongly supported the assumption of temporal correlation theory, but also aroused the interest of investigating relevant computational models for the coherent oscillation phenomenon. Von der Malsburg proposed a neural oscillation model showing that segmentation is expressed by both the synchronization within segments and the desynchronization between segments [4]. After that, many similar models were also investigated. In 2005, Wang [5] surveyed the prominent types of oscillator models: Wilson-Crown, relaxation, and spike oscillators. These models have different characteristics and can accomplish different tasks. For example, the LEGION model by Wang et al. [6] can perform the image segmentation. A contour integration model by Li [7] can mimic the segmentation process of the primary visual cortex. Other kinds of oscillation models have also been designed to fulfill the tasks such as the data clustering [8], the visual attention selection [9], and the audio attention selection [10]. These models revealed the way in which oscillation operates in different tasks

[*] This work is supported by National Basic Research Program of China (No. 2005CB724303), National Natural Science Foundation of China (No. 60671062), and Shanghai Leading Academic Discipline Project (No. B112).

[**] The corresponding author.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 768 – 775, 2009.

especially the ones relevant to machine vision. However, there have not been effective oscillation model that can handle color information well to separate the contours in color images. We propose a neural oscillation model to implement the contour separation for color images. The model can implement both the contour detection and the contour separation by detecting the synchronization and desynchronization of neural oscillations. It can exploit color information rather than just the gray-level information. In Section 2, the model details are described. In Section 3, experiments and results are shown. The discussion and conclusions are given in Section 4.

2 The Neural Oscillation Model

The structure of the model, shown in Figure 1, is similar to the one of the relaxation oscillation model defined in [11]. Each position in an image correlates to a reciprocally connected pair of neurons: the excitatory neuron x_i and the inhibitory neuron y_i. Their dynamic properties are described as

$$\dot{x}_i = f(x_i, y_i) + I_i = 3x_i - x_i^3 + 2 - y_i + I_i \tag{1}$$

$$\dot{y}_i = g(x_i, y_i) = \varepsilon(\alpha(1 + \tanh(x_i / \beta)) - y_i) \ . \tag{2}$$

Here, I_i denotes the external stimulation to the oscillator, and ε, α and β are the oscillation parameters. These are the classical relaxation oscillation equations whose dynamic property was elaborated in [11]. In the equations, the definition of I_i is the key to its capability. If I_i>0, the corresponding oscillators can be stimulated to the active state, otherwise the oscillators will be silent. As a result, I_i is the predominant issue in the oscillation and should be selected carefully according to the required task.

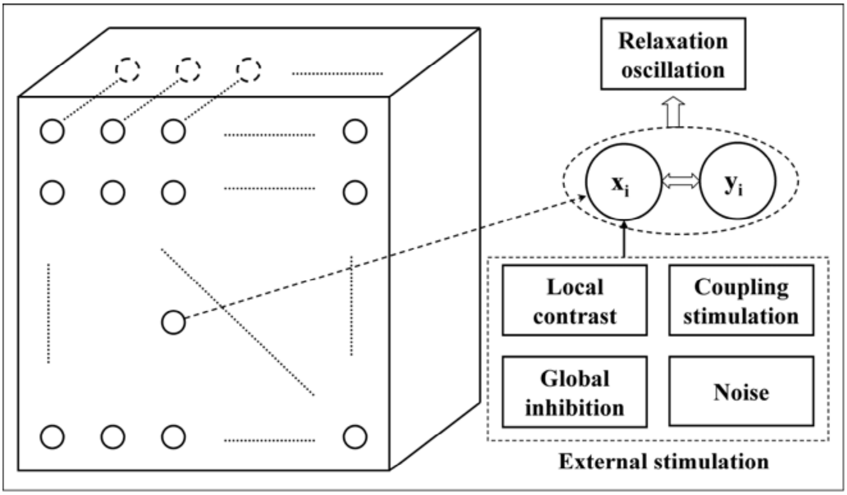


Fig. 1. The structure of the model

2.1 Total External Stimulation

The contour separation task can be seen as a two-step process: detecting the contours of different objects and separating them. From the viewpoint of oscillation, the purpose can be achieved by synchronization and desynchronization, in which the external stimulation I_i plays an important role. We analyze the requirement of contour segmentation and define four kinds of components for I_i. First, the local contrast is a dominant factor for contour detection. The excitatory units in the positions with higher local contrast should be easier to induce oscillation in. Second, there may be several contour elements whose local contrast is low. These elements should also be regarded as belonging to the required objects but they may not be excited only by the local contrast. However, these weak edge elements usually have similar local distribution features to their neighboring strong edge elements. As a result, the coupling stimulus between the edge elements is chosen as another kind of oscillation factor. Two types of coupling are defined here: the color similarity coupling and the orientation continuity coupling. Third, to separate the contours of different objects, desynchronization should be considered. In previous models, desynchronization was achieved by global inhibition from the active neurons. Here the local excitation and the global inhibition are achieved by a uniform global mode. Fourth, the irregular noise was found to be able to facilitate the desynchronizaition in the experiments [12]. Thus it is also considered as one component of the external stimulation. In all, there are four aspects constituting the external stimulation as shown in Figure 1: local contrast S_i, neighboring coupling stimulus C_i, global inhibition Z_i, and noise P_i:

$$I_i=S_i+C_i+Z_i+P_i \,. \tag{3}$$

2.2 Local Contrast

For color images, the local contrast is defined by the average edge strength of three color channels. The Sobel derivative operator is used to filter each channel and then the root-mean-square of the three filtering results is computed. It is labeled as E_i after being normalized to a value between 0 and 1. To ensure the strong edges oscillate and the others do not, there is a threshold T_S to make S_i positive or negative:

$$S_i=E_i-T_S \,. \tag{4}$$

2.3 Neighboring Coupling Stimulus

With only the local contrast, the actual but weak edge elements are given the same treatment as the non-edge elements. To activate them, the coupling stimuli from their nearby strong edge elements are necessary. It means active neurons can send their stimuli to their nearby neurons. The coupling strength between each two neurons depends on their connection probability, which reflects their connectivity and feature similarity.

In Li's model [7], the orientation continuity between edge elements is considered as the main aspect of contour integration. Two edge elements with good connectivity can form an excitatory connection and enhance the activity of each other. As a result, the edge elements belonging to one regular object can have similar oscillation frequency and phase. With this model, the salient contours in one image can be detected by thresholding the average oscillation activity, and then some top-down mechanisms can be used to separate the contours of different objects. Actual results verified that the orientation continuity may play an important role in contour integration. In our model, the orientation continuity is selected as one of the coupling aspects. They are calculated in the following way. The orientation of each position is computed using the orientated DOG (Derivative of Gaussian) filter banks. In detail, the image is filtered by the filter banks with 16 different preferred orientations. The orientation of position *i*, denoted as O_i, is the one that corresponds to the maximum filtering response among all the orientations. In a neighboring area of position *i*, each position *j* has a direction angle D_{ij} against *i*, which is only determined by their relative position. If this direction angle is consistent with the orientation of *j* (denoted as O_j), the position *i* may receive the coupling from *j*. The orientation coupling from *j* to *i* is defined as

$$CO_{ij} = \begin{cases} 0.2/(0.2 + A_{ij}) & if\ A_{ij} \le \pi/6 \\ 0 & if\ A_{ij} > \pi/6 \end{cases}, \tag{5}$$

where A_{ij}=|O_j-D_{ij}|. This formula means that one neuron may receive the coupling stimulus from its neighboring neurons that have good orientation continuity with it. Meanwhile, the coupling strength relies on the orientation continuity.

Except for the orientation continuity, the feature similarity is another aspect of coupling, which is represented by the color similarity between the neighboring areas of two positions. The three-channel color histograms are computed in the neighboring area of each position. Then the distances of the three color histograms between positions *i* and *j* are computed respectively. The root-mean-square of the three values is computed as B_{ij} and the color similarity CC_{ij} is defined as

$$CC_{ij} = 0.2/(0.2 + B_{ij}) \ . \tag{6}$$

The total coupling strength for each position comes from all its neighboring active neurons:

$$C_i = \sum_{j \in N_c(i)} [x_j \cdot (CO_{ij} + CC_{ij}) \cdot \delta(x_j)] \ , \tag{7}$$

where $N_c(i)$ denotes the neighboring area of *i*, and $\delta(x_j)$ is the step function which equals to one for active neurons and zero for the others. A neuron *i* is called 'active' if x_i is larger than the threshold T_a. Finally, the received coupling strength of all the positions is normalized to the range between 0 and 0.5.

2.4 Global Inhibition

With the local contrast and the neighboring coupling stimulus, the condition for the synchronization of oscillation is provided. However, they do not operate in desynchronization. In Wang's model, the locally excitatory and globally inhibitory mechanism was used for desynchronization. Here we design a uniform one for both local excitation and global inhibition:

$$Z_i = e^{-d_{im}^2 / \sigma_z^2} - T_Z \ , \tag{8}$$

where d_{im} is the distance between the position i and the position with the globally maximum oscillation strength, while σ_z and T_z are two parameters determining the range and degree of the excitation or inhibition. In the experiments, σ_z is changed with respect to the image size, while T_z is set as 0.8 to limit Z_i between -0.8 and 0.2.

2.5 Noise

The last issue is the irregular noise P_i. Although it was verified to be able to facilitate the desynchronization, we have not found any obvious effect in our model. It is reserved in the formula to keep the completeness of the oscillation equation but the actual influence of the noise will be investigated in further research.

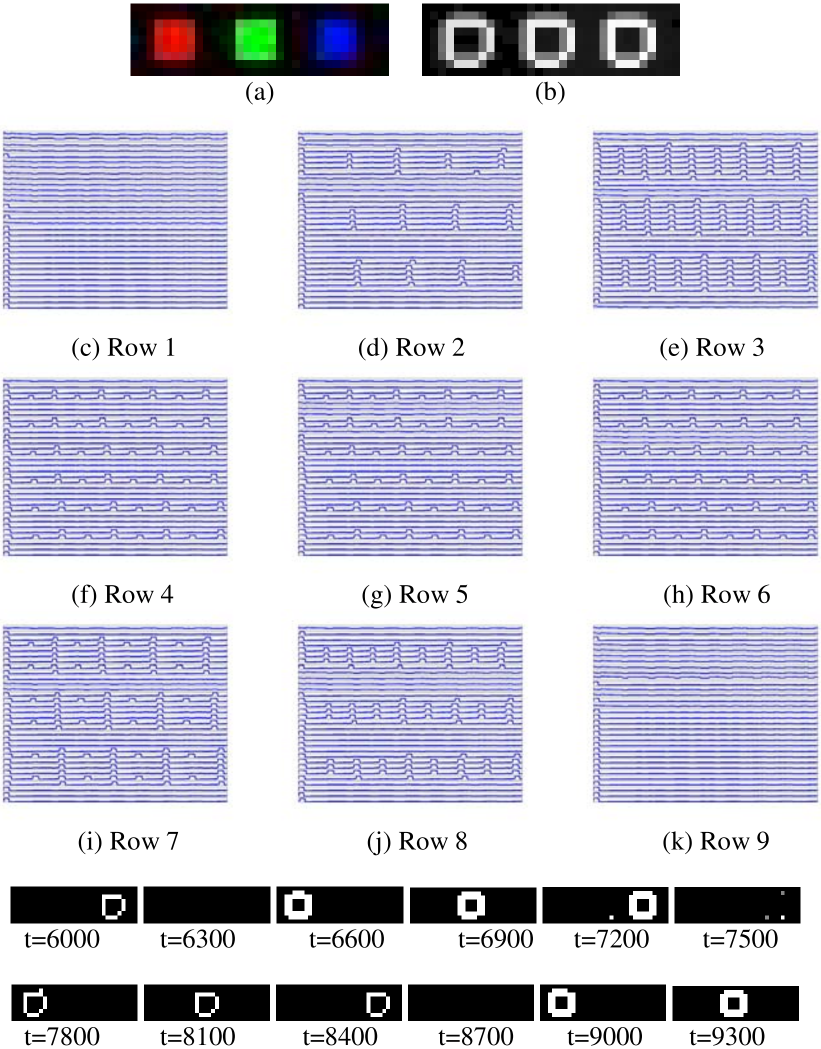
3 Experiments and Results

Although the structure and composition of the model is determined, many parameters still need to be adjusted by analysis and comparison. The parameters not given in the forgoing parts are listed here. In the relaxation oscillation equations, the parameters are set according to previous work, i.e. ε=0.02, α=0.005, β=0.1. Their slight variance does not influence the results much. The edge strength threshold T_s=0.5 and the activity threshold T_a=1.5. The neighborhood is 5-by-5 for coupling and 7-by-7 for calculating the color histograms.

To observe the oscillation situation of the model, several color images with obviously separable objects are used. Figure 2(a) shows a 9-by-32 image which contains three blocks with different colors (red, green, and blue). We use the model and get the average oscillation strength for 10000 iterations, as shown in Figure 2(b). The contours of the blocks have much higher oscillation strength than others, meaning that the model can detect salient contours. Figure 2(c)~(k) show the detailed oscillation process for each position. Each sub-figure corresponds to one row of the original image from the top down. In each sub-figure, the 32 waveforms show the oscillation strength for the 32 points of the corresponding row in the 10000 iterations.

First, the oscillation effect is apparent as seen from the figure: 1) the neurons in the background positions do not oscillate in the entire process; 2) the neurons corresponding to the contours oscillate and have relatively high oscillation frequency; 3) the neurons corresponding to the positions near the contours also have oscillation phenomenon but the frequency is lower.

Second, the synchronization and desynchronization effect can be observed: the positions of one object contour have the similar oscillation phase, while the positions of different object contours have different oscillation phases. For example, it is seen in Figure 2(e)~(j) that the excitation of the green object happens after the excitation of the red one, and before the excitation of the blue one. It should be noticed that the model does not have color preference and the phase is determined by several other aspects such as the initialization and the noise. Another way of observation is shown in Figure 2(l), which depicts the temporary oscillation strengths of the whole image at the iterations of 6000, 6300, 6600, 6900, 7200, 7500, 7800, 8100, 8400, 8700, 9000 and 9300 respectively.



(a) (b)

(c) Row 1 (d) Row 2 (e) Row 3

(f) Row 4 (g) Row 5 (h) Row 6

(i) Row 7 (j) Row 8 (k) Row 9

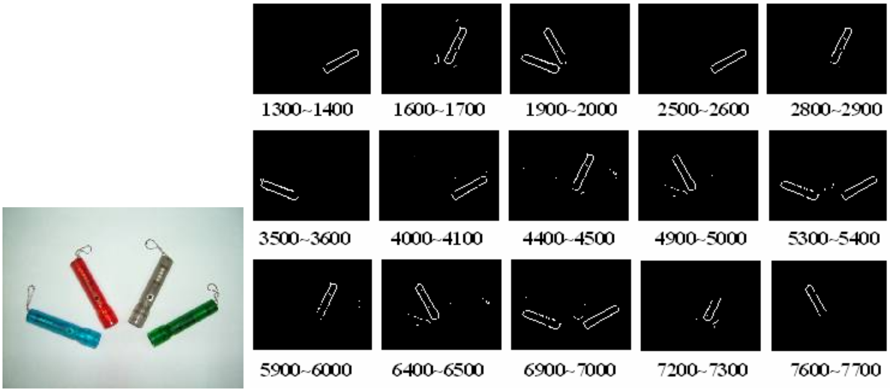
t=6000 t=6300 t=6600 t=6900 t=7200 t=7500

t=7800 t=8100 t=8400 t=8700 t=9000 t=9300

(l)

Fig. 2. One example of the results, (a)original image, (b)average oscillation strength in 10000 iterations, (c)~(k)oscillation results for the nine rows of the original image from the top down, (l)the temporary oscillation strengths at the iterations of 6000, 6300, 6600, 6900, 7200, 7500, 7800, 8100, 8400, 8700, 9000 and 9300 respectively

Another example is shown in Figure 3. The natural image as Figure 3(a) contains four objects with different colors. Figure 3(b) shows the average oscillation strength in different time periods. The selected time periods are representative but not uniformly sampled because the oscillation frequency of each object is not the same. For the same reason, there also exist overlaps between different contours in certain time periods. In spite of these, the synchronization and desynchronization of the oscillation can be observed.



(a) (b)

Fig. 3. One example of the results, (a)original image, (b)average oscillation strength in different time periods

4 Discussion and Conclusions

The proposed neural oscillation model can implement the contour separation for color images. The synchronization and desynchronization of the oscillation are controlled by the external stimulation which comes from the image. The external stimulation comprises four parts: the local contrast, the coupling from neighboring positions, the global inhibition, and the noise. The structure of the model is similar to several previous models. However, this model utilizes the color information reasonably and effectively, conquering the difficulty of investigating the oscillation in color images. The two coupling aspects for oscillation also make it distinct from others.

The model also has some problems to be solved. First, the individual role of the four components of the external stimulation is not yet explicit, especially the role of the noise. We have analyzed the results when each aspect operates individually and found part of the answer, but more experiments are needed to clarify it further. Second, the performance of this model is influenced by the complexity of the image. It performs well in the locally homogenous images but may have a failure when used for complex images. Third, there are many adjustable parameters in the model. The best choice of these parameters is still being explored.

In spite of these problems, the model successfully uses the oscillation in the contour detection and separation for color images. It can help to better investigate both the contour separation methods and the neural mechanisms of it.

Acknowledgements

This work is supported by National Basic Research Program of China (No. 2005CB724303), National Natural Science Foundation of China (No. 60671062), and Shanghai Leading Academic Discipline Project (No. B112).

References

1. Von der Malsburg, C.: The Correlation Theory of Brain Functioning. Internal Report 81-2, MPI Biophysical Chemistry (1981)
2. Eckhorn, R., Bauer, R., Jordan, W., et al.: Coherent Oscillations: A Mechanism of Feature Linking in the Visual Cortex. Biological Cybernetics 60, 121–130 (1988)
3. Gray, C.M., Konig, P., Engel, A.K., Singer, W.: Oscillatory Responses in Cat Visual Cortex Exhibit Inter-columnar Synchronization Which Reflects Global Stimulus Properties. Nature 338, 334–337 (1989)
4. Von der Malsburg, C., Schneider, W.: A Neural Cocktail-party Processor. Biological Cybernetics 54, 29–40 (1986)
5. Wang, D.L.: The Time Dimension for Scene Analysis. IEEE Trans. Neural Networks 16, 1401–1425 (2005)
6. Wang, D.L., Terman, D.: Locally Excitatory Globally Inhibitory Oscillator Networks. IEEE Transactions on Neural Networks 6, 283–286 (1995)
7. Li, Z.P.: A Neural Model of Contour Integration in the Primary Visual Cortex. Neural Computation 10, 903–940 (1998)
8. Rhouma, M.B.H., Frigui, H.: Self-organization of Pulse-coupled Oscillators with Application to Clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence 23, 180–195 (2001)
9. Kazanovich, Y., Borisyuk, R.: Object Selection by an Oscillatory Neural Network. BioSystems 67, 103–111 (2002)
10. Wrigley, S.N., Brown, G.J.: A Computational Model of Auditory Selective Attention. IEEE Transactions on Neural Networks 15, 1151–1163 (2004)
11. Terman, D., Wang, D.L.: Global Competition and Local Cooperation in a Network of Neural Oscillators. Physica D 81, 148–176 (1995)
12. Freeman, W.J.: Neurodynamics: An Exploration in Mesoscopic Brain Dynamics. Springer, London (2000)

A Color Image Segmentation Using Inhibitory Connected Pulse Coupled Neural Network

Hiroaki Kurokawa, Shuzo Kaneko, and Masato Yonekawa

Tokyo University of Technology, 1404-1 Katakura Hachioji Tokyo, Japan
hkuro@bs.teu.ac.jp

Abstract. A Pulse Coupled Neural Network (PCNN) is a kind of numerical model of cat visual cortex and it can explain synchronous dynamics of neurons' activity in the visual cortex. On the other hand, as an engineering application, it is shown that the PCNN can applied to the image processing, *e.g.* segmentation, edge enhancement, and so on. The PCNN model consists of neurons and two kind of inputs, namely feeding inputs and linking inputs with leaky integrators. These inputs lead to discrete time evolution of its internal state and neurons generate spike output according to the internal state. The linking and feeding inputs are received from the neurons' receptive field which is defined by excitatory synaptic weights. In this study, we propose a PCNN with inhibitory connections and describe an application to a color image segmentation. In proposed model, inhibitory connections are defined by negative synaptic weights among specific neurons which detect RGB component of particular pixel of the image. Simulation results show successful results for the color image segmentation.

1 Introduction

A Pulse Coupled Neural Network (PCNN) is proposed as a model which can explain synchronous dynamics of neurons' activity in cat visual cortex[1]. The PCNN model consists of neurons and two kind of inputs, namely feeding inputs and linking inputs with leaky integrators. These inputs lead to discrete time evolution of neurons' internal state and neurons generate spike output corresponding to the internal state. The linking and feeding inputs are received from neurons' receptive fields which is defined by synaptic weight and directly from the environment.

On the other hand, from the engineering point of view, PCNN is considered as a temporal pulse coding system which shows a synchronous pulse dynamics in a network and it is known that the most significant characteristics of PCNN is its rich power of pulse coding expression. A lot of applications to the engineering field are proposed, especially in the field of image processing, *e.g.* segmentation, edge enhancement, and so on[2][3][5]. Also, hardware implementation of PCNN is described in recent study[6] and digital spike neuron[7][8], PCNN like system, which consists of shift-register array had been proposed. As shown in these studies, It has been expected that an application of PCNN scheme broads to the various fields of engineering.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 776–783, 2009.

A pattern segmentation is one of the most significant issue in the image processing and an application of the PCNN for pattern segmentation is proposed[2]. To achieve a pattern segmentation, 2D-PCNN which has latticed connection has been used and one neuron in the network corresponds to one pixel of the image to be processed. In conventional studies, the target image is assumed as a gray-scale image, namely, the neuron in the PCNN receives a intensity value of a pixel of the image.

A color image processing using PCNN is also studied recently[9][10][11]. In these studies, a sampling component of color image, *e.g.* intensity, saturation, hue, is used as a external input to PCNN. In other words, some information of color component is lost through the processing. Also, to achieve pattern segmentation of color images, Multi-channel PCNN system had been proposed[12]. This enhanced model for color image processor consists of plural PCNNs and extra processing unit. However, the system requires decision of particular target color according to the image and external calculation unit for image processing.

In this study, we propose a PCNN with inhibitory connections and describe an application to color image segmentation. Our proposed Inhibitory Connected PCNN(IC-PCNN) has three types of neuron. We assume that each type of neuron can detect RGB component of the image as observed in vivo retinal cone cell[13]. In our proposed model, inhibitory connections are defined by negative synaptic weight among specific neurons which detect RGB component of same pixel of the image. Where we assume that the spike inputs via inhibitory connections inhibit R-, G-, and B-type neuron each other according to its negative synaptic weight. In this paper, We also show an implementation of our proposed IC-PCNN to the color image segmentation. Most remarkable issue of our proposed IC-PCNN is that the IC-PCNN requires no additional system which cannot be observed in vivo system to achieve color image processing.

2 PCNN Model

2.1 Conventional Model of PCNN

PCNN is a model of cat visual cortex proposed by Echorn *et.al.*[1] and a lot of applications in engineering field has been proposed especially in image processing [2][3][4]. Figure 1 shows a schematic of PCNN. The model consists of the dendrite model and the soma model. In the PCNN, dendrite model forms connections among neurons and input from an environment and soma model is functioning as a spike generator.

In general, the PCNN for the image processing has two dimensional structure with lattice connection among neurons and each neuron in the PCNN receives information from each corresponding pixel via feeding input. The two dimensional PCNN model is mathematically described as follows. The internal state of the neuron ij, namely membrane potential in biological model, is given by,

$$U_{ij}(t) = F_{ij}(t)(1 + \beta_{ij} L_{ij}(t)). \tag{1}$$

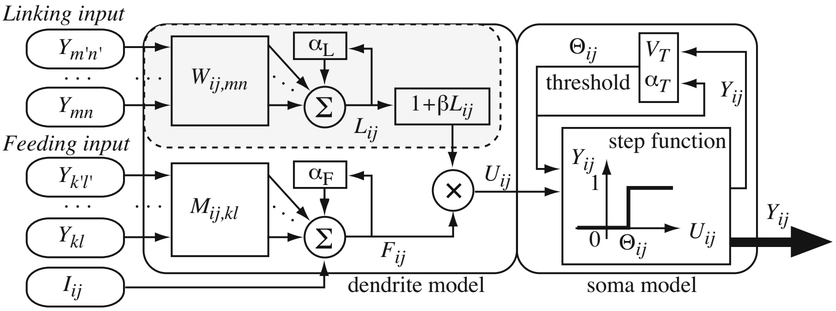


Fig. 1. The schematic model of Pulse Coupled Neural Network (PCNN)

Note that the indices i, j denote neuron number in the two dimensional PCNN. In the PCNN model, there are two kinds of inputs which are propagated via different connections. One is a *feeding input* and the other is a *linking input*. Each input is described as follows, respectively.

$$F_{ij}(t+1) = F_{ij}(t)\exp\left(-1/\alpha_F\right) + V_F \sum_k \sum_l M_{ij,kl} Y_{kl}(t) + I_{ij}, \tag{2}$$

$$L_{ij}(t+1) = L_{ij}(t)\exp\left(-1/\alpha_L\right) + V_L \sum_m \sum_n W_{ij,mn} Y_{mn}(t). \tag{3}$$

Where $M_{ij,kl}$ and $W_{ij,mn}$ are synaptic weights which define a receptive field of the neuron, I_{ij} is constant inputs to the neuron, and $Y_{kl}(t)$ and $Y_{mn}(t)$ are spike output of neuron kl and mn, respectively. This spike output is defined as a step function which is given by,

$$Y_{kl}(t) = \begin{cases} 1 & \text{if} \quad U_{kl}(t) > \Theta_{kl}(t) \\ 0 & \text{else} \end{cases} . \tag{4}$$

In Eq.(4), $\Theta_{kl}(t)$ is a threshold of the action potential of the neuron kl which is given by,

$$\Theta_{kl}(t+1) = \Theta_{kl}(t)\exp(-1/\alpha_T) + V_T Y_{kl}(t) \tag{5}$$

Through Eq.(1)−Eq.(5), parameters, β_{ij}, α_F, α_L, α_T, V_F, V_L, and V_T are decided appropriately.

2.2 Introducing Inhibitory Connections to the PCNN

In this study, we extend *a "monochrome PCNN"* to *a "color PCNN"* by introducing inhibitory connections. To achieve color image processing, we first assumed that the *"color PCNN"* has three neurons per pixel and each neuron detect each RGB component of the pixel color. It is known that in vivo retinal cone cell can detect specific wavelength of lightwaves. Actually, human retinal cone cell shows three types of spectral response and each of them can detect one of RGB components of the color[13]. Therefore, it is considered that the structure of the *color PCNN* is not contrary to biological assumptions.

Further, we assumed that the *"color PCNN"* has inhibitory connections among neurons which correspond to the same pixel. Where inhibitory connections are defined as follows.

$$\boldsymbol{\Gamma}_{ij} = \begin{pmatrix} 0 & \gamma_{GR} & \gamma_{BR} \\ \gamma_{RB} & 0 & \gamma_{GB} \\ \gamma_{RG} & \gamma_{BG} & 0 \end{pmatrix} . \tag{6}$$

Where the parameters γ has negative value, and all the parameters of γ is defined appropriately to lead effective characteristics of the color PCNN for the application, such as segmentation, edge detection, and so on.

Based on these assumptions of inhibitory connection, in this study, we propose an Inhibitory Connected Pulse Coupled Neural Network (IC-PCNN) for color image segmentation. The schematic model of proposed IC-PCNN is illustrated in Figure 2. The feeding input and linking input of proposed IC-PCNN is given by,

$$\begin{aligned} \begin{pmatrix} F_{ij,R}(t+1) \\ F_{ij,G}(t+1) \\ F_{ij,B}(t+1) \end{pmatrix} &= \begin{pmatrix} F_{ij,R}(t) \\ F_{ij,G}(t) \\ F_{ij,B}(t) \end{pmatrix} \exp(-1/\alpha_F) + V_F \sum_{k,l} M_{ij,kl} \begin{pmatrix} Y_{kl,R}(t) \\ Y_{kl,G}(t) \\ Y_{kl,B}(t) \end{pmatrix} \\ &+ V_F \boldsymbol{\Gamma}_{ij} \begin{pmatrix} Y_{ij,R}(t) \\ Y_{ij,G}(t) \\ Y_{ij,B}(t) \end{pmatrix} + \begin{pmatrix} I_{ij,R} \\ I_{ij,G} \\ I_{ij,B} \end{pmatrix} , \end{aligned} \tag{7}$$

$$\begin{aligned} \begin{pmatrix} L_{ij,R}(t+1) \\ L_{ij,G}(t+1) \\ L_{ij,B}(t+1) \end{pmatrix} &= \begin{pmatrix} L_{ij,R}(t) \\ L_{ij,G}(t) \\ L_{ij,B}(t) \end{pmatrix} \exp(-1/\alpha_L) + V_L \sum_{m,n} W_{ij,mn} \begin{pmatrix} Y_{mn,R}(t) \\ Y_{mn,G}(t) \\ Y_{mn,B}(t) \end{pmatrix} \\ &+ V_L \boldsymbol{\Gamma}_{ij} \begin{pmatrix} Y_{ij,R}(t) \\ Y_{ij,G}(t) \\ Y_{ij,B}(t) \end{pmatrix} . \end{aligned} \tag{8}$$

Where $F_{ij,R}$, $F_{ij,G}$, $F_{ij,B}$,$L_{ij,R}$, $L_{ij,G}$, and $L_{ij,B}$ is feeding or linking input of R-type, G-type, and B-type neuron of pixel ij. Also, we assume that the internal state of the neuron ij is given by,

$$U_{ij}(t) = \begin{cases} F_{ij}(t)(1 - \beta_{ij} L_{ij}(t)) & \text{if} \quad F_{ij} < 0 \quad \text{and} \quad L_{ij} < 0 \\ F_{ij}(t)(1 + \beta_{ij} L_{ij}(t)) & \text{else} \end{cases} . \tag{9}$$

3 Simulation Results

3.1 Pattern Segmentation of Color Test Images

In this section, we show simulation results of our proposed IC-PCNN. We first show the results of the segmentation of color test images. Here, we use an Ishihara

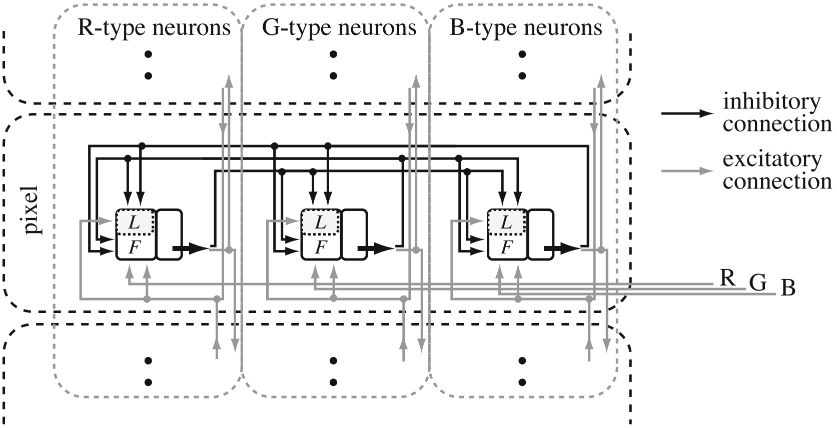


Fig. 2. The schematic model of Inhibitory Connected Pulse Coupled Neural Network (IC-PCNN) is illustrated. The IC-PCNN has neurons for each RGB component and inhibitory connections are defined among neurons in same pixel.

color test[14][15] that is used in the test of color blindness. Figure 3 shows a sample image of the color test[1] which indicates a character ”74” in the circle. However, as shown in Figure 3(a), we cannot recognize a character ”74” in the circle from the gray-scaled image because the intensity difference of each pixel in the circle of the image does not form the character ”74”. Each of RGB component of this image is shown in Figure 3(b) (c) (d), respectively. From these image, we can find that the red component image indicate the character ”74” comparatively to the other component images, but the image doesn't indicate the character obviously. From this peculiar characteristics of the color test image, it is clearly considered that conventional PCNN method concerning an intensity of the image cannot achieve segmentation of this character. This estimate is confirmed in Figure 4.

Then, we try to achieve a segmentation of the test image by using each RGB component image which is shown in Figure 3(b) (c) (d). The simulation results are shown in Figure 5. As shown in Figure 5, almost of results does not show successful segmentation. In the case that the red component of the test image is used, which is shown in Figure 5(upper row), we can find a segmented character ”74” *comparatively* to the other trials. However, to obtain this result of segmentation, a PCNN requires additional prior information which color component of the image shows characteristics of the pattern that is to be segmented. Then we can conclude that the segmentation of the color test image is hardly to achieve using conventional PCNN.

[1] Here we cannot show an actual color image of the color test because this manuscript will be published in monochrome. However, we can easily find this color test on webpage[15].

Figure 6 shows the results of segmentation by proposed IC-PCNN. Target image of segmentation is also shown in Figure 3, and all of the RGB component of the image, that are shown in Figure 3(b)(c)(d), are used as inputs to the IC-PCNN. The simulation results show that the segmentation of the character ”74” is indicated in several time step. Here, considering that the proposed IC-PCNN requires no additional processing unit and artificially decided information from images to achieve segmentation, we conclude that the IC-PCNN has the qualities for the color image segmentation essentially.

Note that, through the simulations, a pixel size of the picture shown in Figure 3(a) is 128×128, the conventional PCNN has 128×128 neurons and connections, and the proposed IC-PCNN has 128×128×3 neurons and connections.

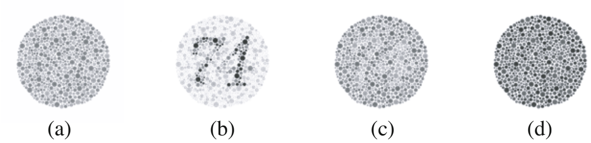


Fig. 3. Ishihara color test: images are color images in reality. In this figure, we show (a)gray scale image, (b) red component of the image, (c) green component of the image, and (d) blue component of the image. Purpose of the segmentation is to make a segmentation of character ”74”.

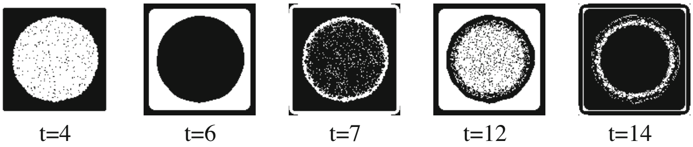
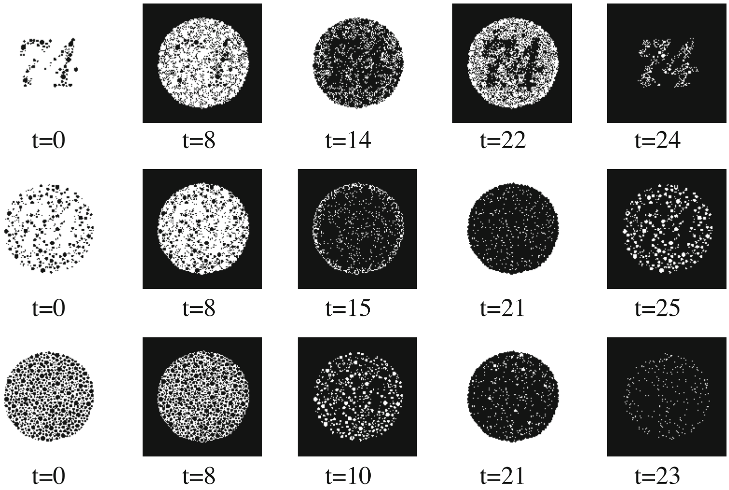


Fig. 4. Simulation results of the segmentation by a conventional method using PCNN. The input image is gray-scaled image (only intensity of the image is considered) which is shown in fig.3 (a). No characters or significant segments are shown in any time step.

3.2 Pattern Segmentation of Photo Images

In previous section, we show the results of peculiar test image. Then, in this section, we show an example of color image segmentation using relatively common image. Figure 7(upper row, left) shows the input image to be segmented and the image shows four deferent color files. As shown in Figure 7, proposed IC-PCNN achieves segmentation of all color files in several time steps. Also in these simulations, IC-PCNN does not require artificially decided information except that the parameters for PCNN. Therefore IC-PCNN is able to work as a complete system for pattern segmentation of color images.

Note that, in the simulation, a pixel size of the picture shown in Figure 7 is 100×60 and the IC-PCNN has 100×60× 3 neurons and connections.

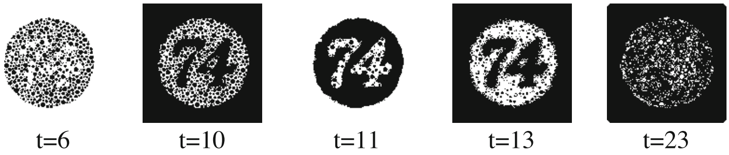


t=0 t=8 t=14 t=22 t=24

t=0 t=8 t=15 t=21 t=25

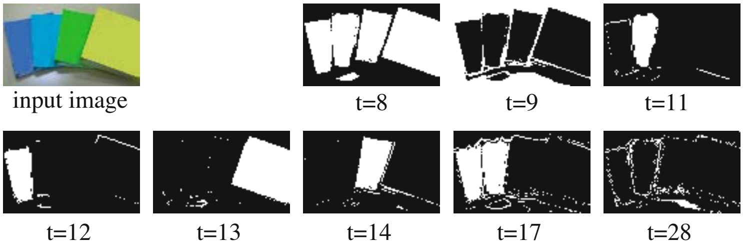
t=0 t=8 t=10 t=21 t=23

Fig. 5. Simulation results of the segmentation by a conventional method using PCNN. Upper row: the input image is red component of the image which is shown in Figure3 (b). Middle row: the input image is green component of the image which is shown in Figure 3 (c). Lower row: the input image is blue component of the image which is shown in Figure 3 (d). The target character "74" is *"comparatively"* segmented by using red component input image, but *"obvious"* segmentation does not achieved in any input images.



t=6 t=10 t=11 t=13 t=23

Fig. 6. Simulation results of the segmentation by our proposed IC-PCNN. The input image is full color image. The target character "74" is segmented as shown in several time steps.



input image t=8 t=9 t=11

t=12 t=13 t=14 t=17 t=28

Fig. 7. Test image and simulation results of the segmentation by our proposed IC-PCNN. Upper row, left: Deferent color of four files. The input image is full color image in reality. Each color is violet, green, blue, and yellow. Upper row, right and lower row: Simulation results of the segmentation by the proposed IP-PCNN. Each file is segmented in several time steps.

4 Conclusion

In this study, an Inhibitory Connected Pulse Coupled Neural Network (IC-PCNN) is proposed and its application for the color image segmentation is presented. Our proposed IC-PCNN requires no extra processing unit in the system and a component of IC-PCNN is not contrary to biological assumptions. In other words, the IC-PCNN is considered as a biologically plausible system. We showed successful simulation results of color image pattern segmentation by using IC-PCNN, while it cannot be achieved by conventional methods using PCNN. In this study, an appropriate parameters were empirically decided in simulations, and the problem of parameter optimization will be solved in our future works.

References

1. Echorn, R., Reitboeck, H.J., Arndt, M., Dicke, P.: Feature linking via synchronization among distributed assemblies: Simulations of results from cat visual cortex. Neural Computation 2, 293–307 (1990)
2. Echorn, R.: Neural Mechanisms of Scene Segmentation: Recording from the Visual Cortex Suggest Basic Circuits for Liking Field Model. IEEE Trans. Neural Network 10(3), 464–479 (1999)
3. Johnson, J.L., Padgett, M.L.: PCNN Models and Applications. IEEE Trans. Neural Network 10(3), 480–498 (1999)
4. Lindblad, T., Kinser, J.M.: Image processing using Pulse-Coupled Neural Networks, 2nd edn. Springer, Heidelberg (2005)
5. Ogawa, Y., Ishimura, K., Wada, M.: Stereo Vision using Pulse-Coupled Neural Network. In: Proc. of SICE, pp. 719–724 (2002)
6. Vega-Pineda, J., Chacon-Murguia, M.I., Camarillo-Cisneros, R.: Synthesis of Pulse-Coupled Neural Networks in FPGAs for Real-Time Image Segmentation. In: Proc. of IJCNN, pp. 8167–8171 (2006)
7. Torikai, H., Saito, T.: Digital spiking neuron and its approximation ability of spike-trains. IEICE Technical Report, NLP2007-1, pp.1–6 (2007)
8. Torikai, H.: Fundamental analysis of a Digital Spiking Neuron for development of a learning algorithm. IEICE Technical Report, NC2007-60, pp. 31–36 (2007)
9. Gu, X.: A New Approach to Image Authentication using Local Image Icon of Unit-Linking PCNN. In: Proc. of IJCNN, pp. 2015–2020 (2006)
10. Zhou, L., Sun, Y., Zheng, J.: Automated Color Image Edge Detection using Improved PCNN Model. WSEAS Transactions on Computers 7(4), 184–189 (2008)
11. Xiong, X., Wang, Y., Zhang, X.: Color Image Segmentation using Pulse-Coupled Neural Network for Locusts Detection. In: Proc. of the International Conference on Data Mining, pp. 410–413 (2006)
12. Sakai, T., Tokito, K., Ogawa, Y., Ishimura, K., Wada, M.: Visual Processing Based on Pulse Coupled Neural Network and Applications. In: Proc of SSI 2002, pp. 381–386 (2002)
13. Alberts, B., et al.: Molecular Biology of the Cell. Garland Science (2001)
14. Ishihara, S.: Tests for colour-blindness. Handaya (1917)
15. Images of the Ishihara color test are able to be found on webpages, for example, http://en.wikipedia.org/wiki/Ishihara_color_test, http://www.toledo-bend.com/colorblind/Ishihara.html, etc

Generating Saliency Map Related to Motion Based on Self-organized Feature Extracting

Satoru Morita

Faculty of Engineering, Yamaguchi University

Abstract. A computational theory concept generating saliency maps from feature maps generated in the bottom-up using various filters such as Fourier transformation was discussed. We proposed saliency map related to motion based on self-organized feature extracting not using general filter such as Fourier transform. We introduce the ICA base function to realize the self-organized Saliency Map. We extend the ICA base function estimation to apply for the non-uniform positioned photoreceptor cells which receives the current image and the previous image to get the motion information. We show the effectiveness of our model by applying this model for real images.

1 Introduction

Marr showed a calculation model of the initial vision that extracts a feature in the bottom-up from the image[1]. Koch and Ullman proposed a computation model computing the saliency map which shows a saliency in two-dimensional plane from the feature map generated in the bottom-up[2]. Itti and Koch discussed the computation model which can be applied to the image analysis[3]. The filter is the feature extraction of the multiresolution, and it is constructed using general Fourier transformation and a rotation filter. Because these use a more general filter, it is necessary for the image to be uniformly arranged. Fourier transformation and Wavelet were proposed to extract a feature in the bottom-up from the image[4]. The information related to the frequency by convoluting a sine wave in the original signal can be gotten it by the Fourier transformation. A base function becomes a sine wave here. The information related to the frequency by convoluting a mother base function in the original signal can be gotten it using Wavelet. The technique is based on the base function. On the other hand, the method how to extract a feature signal by estimating a ICA base[5] was discussed. ICA is an abbreviation of the independent component analysis. An adaptation example is application to the signal and the image arranged uniformly. In this paper, the defined general filter such as Fourier transform isn't used, but the self-organizing filter generated only from the observation of the image is used. Therefore the general idea of the ICA base estimation is introduced in the model which generates a saliency map. The receptive field of the human vision has the arrangement which isn't especially uniform. For example, the density of the receptive field is high in the center, and is low in the

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 784–791, 2009.

circumference with a foveated vision. Our model is expanded so that the filter with the receptive field has a non-uniform arrangement like human.

Hubel and Wiesel discovered that there was a neuron which reacts to the slit light of the specifical inclination in the visual cortex of the cerebral membrane of the cat[6] . Such a cell is called a feature extracting cell. The cells which react to the optical motion are observed. Blakemore and Cooper found that the feature extracting cell who responded to the horizontal slit light was not formed, but the feature extracting cell which responded to the vertical slit light was formed[7] . Hubel and Wiesel explained how a discovered neuron was arranged with Malsburg by the neural network model that a Hebb learning was adopted[8]. Willshaw and Malsburg showed the structure that the combination of the topological mapping seen in the living body was formed by learning by using the neural network model[9]. Kohonen expanded the interpretation of the neural network model of the topological mapping from the position of the information processing[10]. The function of the vision system surveyed here is realized in this paper.

The self-organizing filter which can be computed to the receptive field which has such arrangement which isn't uniform is generated from only observing. When a saliency map is specially generated, it pays attention to the important motion, and a initial vision is modeled. This problem of motion was not discussed in saliency map[2][3].

2 A Requirement for the Vision

[1] The feature which is dependent on the input conditions should be detected.
[2] The input sensor arrangement of the position which is not uniform should be allowed such as retina.
[3] The information that the retina receives should be only analyzed.
[4] The feature should be detected without depending on the order observing the image.

To realize the requirement [1], we introduce ICA base function estimation to generating the saliency map. To realize the requirement [2], our model is expanded the input for the ICA base function estimation can be applied to the arrangement which isn't uniform as shown in figure 1(a) and (b). To realize the requirement [3], information to receive is restricted to the minimum information which a living body receives. The requirement [4] can be realized by the feature extraction based on ICA.

[5] The motion should be distinguished.
[6] An edge should be distinguished.
[7] A texture should be distinguished.

We make the receptive field to receive the local intensity information which is invariant for the position and the direction to realize the requirement [5]. We make the receptive field to receives the neighbor information which is needed for the texture analysis and the edge detection at the same time to realize the requirements [6] and [7]. It tries so that receptive field is arranged spatially.

[8] The various filters of a person's living body should be generated.

The requirements [8] is discussed after a result of an experiment.

3 Saliency Map and Self-organized Filters Based on ICA Base Function Estimation

The method that an ICA base function is estimated from the image and the signal was discussed. The self-organization of the feature extracting cell of the motion are modeled based on the ICA base function estimation in this paper. The feature is detected such that the data projected to the subspace is independent and distributed. When the data that $X_l = \{x_{l1}, x_{l2}, \cdots, x_{ln}\}$ is centering, Y_l is defined in the following as

$$Y_l = W_l X_l, \tag{1}$$

where $Y_l = \{y_{l1}, y_{l2}, \cdots, y_{ln}\}$ and $W_l = \{w_{l1}, w_{l2}, \cdots, w_{ln}\}$. The saliency map and the ICA base function are estimated for the lth data X_l. The independence $J0_l$ of Y_l is evaluated as

$$J0_l = \Sigma_{i,j(i \neq j)} E\{y_{li} \cdot y_{lj}\}. \tag{2}$$

The degree of the distribution $J1_l$ of Y_l is evaluated as

$$J1_l = \frac{1}{\Pi_i E\{y_{li} \cdot y_{li}\}}, \tag{3}$$

where,

$$|w_l i| = 1(l = 1, \cdots, m, i = 1, \cdots, n). \tag{4}$$

The general evaluation J_l is defined as

$$J_l = k_0 \cdot J0_l + k_1 \cdot J1_l, \tag{5}$$

where $J_l(l = 1, \cdots, m)$ which is saliency map value based on the J value of the m individual. It returns in the problem which W_l is decided as so that J_l may become the smallest. It becomes the base function which W_l is estimated at. It is solved by the method of steepest descent.

When the input data that X_l equals $\{x_{l1}, x_{l2}, \cdots, x_{ln}\}$ is centering, Y_l equals $\{y_{l1}, y_{l2}, \cdots, y_{ln}\}$ and W_l equals $\{w_{l1}, w_{l2}, \cdots, w_{ln}\}$. The feature Y_l of the n individual is extracted from the input data of the n individual using equation (1). The salience map is defined based on the feature value $Sa_l = max\{y_{l1}, \cdots, y_{ln}\}$ and the feature number $Sb_l = \Sigma_{i=1}^{n} f(y_{li})$, where a function f used step function. A saliency map based on the feature value is generated by using $Sa_l(l = 1, \cdots, m)$. A saliency map based on the number of features is generated by using $Sb_l(l = 1, \cdots, m)$. The parameter k_0 and k_1 are defined as $k_0 = 1$ and $k_1 = 0$ simply not using distribution but using only independency. It is known that the arrangement of the receptive field is the circle and the rectangle and so on. Generally it is seldom arranged by the receptive field of the biological system uniformly. The receptive field is arranged according to the probability distribution which is dense in the center and is sparse in the periphery. Because the input sensor position of the image is in two dimensions, a probability density function is defined in the two dimension, too. The receptive field is input generally. If the number of the receptive field is 10, n in X which equals $\{x_1, x_2, \cdots, x_n\}$ becomes 10. In a cell to react to the motion, n is $10 \cdot 2 = 20$ because a cell receives two values from the intensity images at a current time and and a previous time.

4 Generating Saliency Map Related to Motion

An optical flow is proposed to detect the motion of the light in the computer vision field[11]. The intensity of the current time and the intensity of the previous time are at least necessary to calculate an optical flow. The inputs which receptive field receives are the intensity of the current time and the intensity of the previous time to interpret the motion of the light. A photoreceptor receives 18 inputs. The probability of that arrangement that receptive field exists in the center is high, and probability in the periphery lowers. Figure 1(a) and (b) show photoreceptors for the previous image in a layer and the current image in another layer to detect motion. It was arranged in accordance with the probability distribution of figure 1(c). This brain model is the process between eye to vision first field as shown in figure 1(d). Figure 2 shows the flow from the photoreceptor which reacts to the motion to the primary visual cortex.

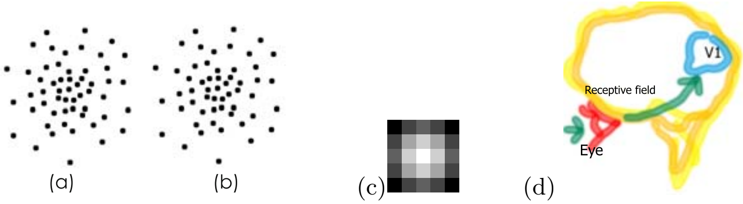


Fig. 1. Two layers photoreceptors to detect motion. (a) A photoreceptor for the previous image. (b) A photoreceptor for the current image. (c) The probability distribution of the receptive field of the foveated vision (d)This brain model is the process between eye to vision first field.

- A photoreceptor receives the intensity of the current time and the intensity of the previous time.
- A feature by multiplying the estimated base function in the intensity of the previous time and the current time is gotten.
- Saliency map value is calculated from the feature value and the number of features.
- Event map is calculated from competitive learning of the detected feature value.

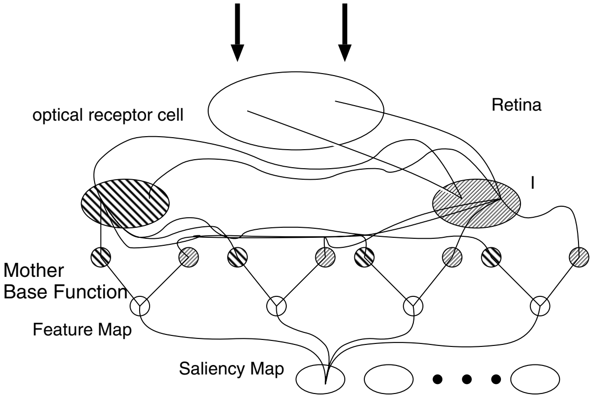


Fig. 2. Flow from the photoreceptor which reacts to the motion to the primary visual cortex

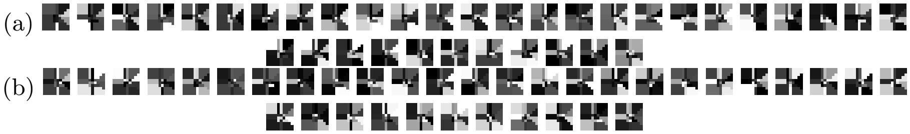


Fig. 3. (a) 36 base functions for the intensity of the current time. (b) 36 base functions for the intensity of the previous time.

Figure 5(a) and 5(b) shows two sheets of images such as laboratory including chairs and desks. The image size is 512 × 512. The next viewpoint is decided using the probability from the whole of the image simply in this method. The receptive cell receives 18 inputs from the intensity images of the current time and the previous time. Figures 3 (a) and 3 (b) show 36 base functions of the receptive cells which receive the intensity of the current time and the previous time. Figure 4 shows 36 feature images. The feature extracting cells which are black in the center, and white in the periphery are found. On the other hand, the feature extracting cells which are white in the center, and black in the periphery are found. The feature extracting cells which are white only in a

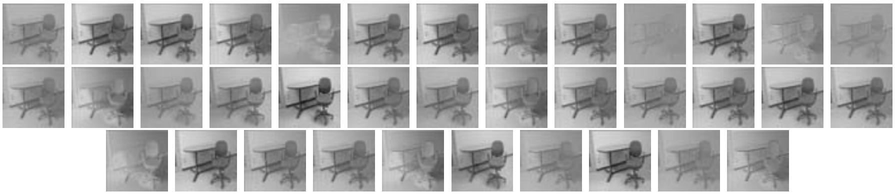


Fig. 4. 36 feature images

certain direction are found. The various filters which a person's living body has should be generated. The requirement [8] is satisfied in the experiment. There are relations as for the two images in the neighbor. The base function is the reverse of a certain base function. These cells are seen by the receptive field of the ganglion cell and the primary visual cortex. There are relations in a cell to react to the stimulus to move to an on-center off-surround form cell, a off-center on-surround form cell and the cell of the orientation sensitivity and the specifical direction which it has strongly. A more remarkable base function becomes clear by increasing the number of the receptive cell. It is meaningful that the self-organizing cell which is common with the actual receptive field is generated. Figures 5 (c) and 5 (d) show a saliency map based on the feature value and the number of features respectively. Figure 5 (e) shows the event map classified features using competitive learning. From figures 5 (c) and 5 (d), it is found that the part which changes from the light place to the dark place , and the part which changes from the dark place to the light place is classified. It is found that many feature extracting cells which perceive a change of the brightness in the image exist.

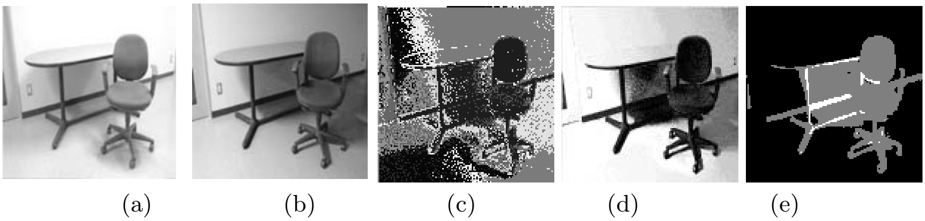


Fig. 5. (a) (b) Two images which receptive field receives. (c) A saliency map based on the feature value. (d) A saliency map based on the feature number. (e) An event map classified the detected features.

Figures 8(a) and 8(b) show the image of driver's view which receptive field receives. The method which the next viewpoint is decided using the probability from the whole of the image and so on was used for simply here. A receptive cell receives 12 inputs from the intensity images of the current time and the previous time. Figures 6 (a) and 6 (b) show 24 base functions of the receptive cells which receive the intensity of the current time and the previous time. Figure 7 shows 24 feature images. Figures 8 (c) and 8 (d) are based on each of the feature value and the number of features. Figure 8 (e) shows the event map classified features using competitive learning. From figures 8 (a), 8 (b) and 8 (c), it is found that

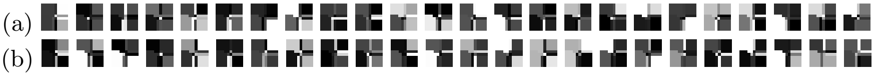


Fig. 6. (a) 24 base functions for the intensity of the current time. (b) 24 base functions for the intensity of the previous time.

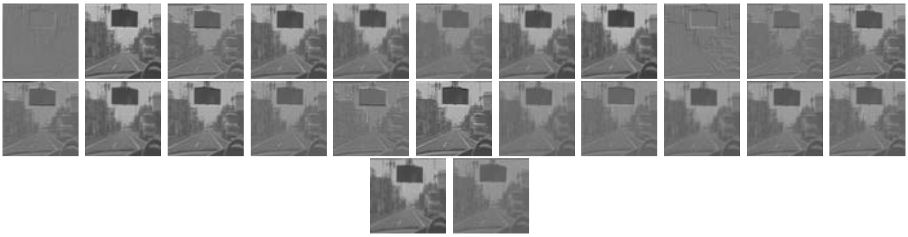


Fig. 7. 24 feature images

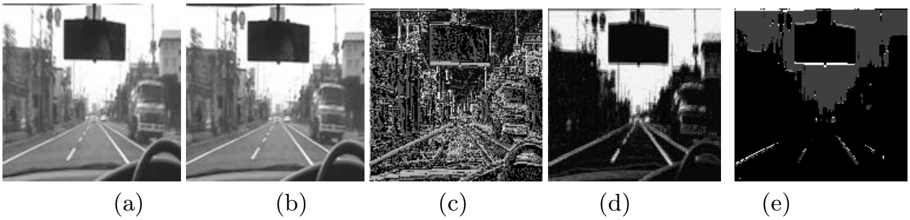


Fig. 8. (a) (b) Two images which receptive field receives. (c) A saliency map based on the feature value. (d) A saliency map based on the feature number. (e) An event map classified the detected features using competitive learning.

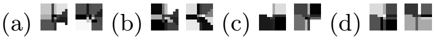


Fig. 9. Base functions of (a) and (b) to react to the motion are included in figure 3(a) and (b). Base functions of (c) and (d) to react to the motion are included in figure 6(a) and (b). Two base functions of the current time and the previous time are shown.

the part of the sky and the road line and the part of moving cars, road signs and townscape are classified. It is found that many feature extracting cells which perceive a change in the brightness in the image exist from figure 6. Base functions of figures 9 (a) and 9 (b) to react to the motion are included in figures 3(a) and 3(b). Base functions of figures 9 (c) and 9 (d) to react to the motion are included in figures 6(a) and 6(b). Two base functions of the current time and previous time are shown. The black changes white and the white changes black at a time in the base function to react to the motion. The high level process following with the low level process and the event analysis of the task can be managed because the classification image of a motion is similar to the event map based on the detected features. The input which receptive field receives is made intensity, and the intensity of the previous time here to interpret the motion of the light. The motion is detected by two layers that a layer of photoreceptor receives a previous image and another layer of photoreceptor receives a current image as shown in figures 1(c) and 1(d).

5 Conclusion

The self-organization of the feature extracting cell which reacts to the motion of the local intensity was modeled based on the ICA base function estimation. The arrangement of the receptive field isn't uniform, and the various remarkable cells is generated by a self-organization filter. It is an effectiveness as a model because the method is simple.

References

1. Marr, D.: VISION. WHF Freeman and Company, New York (1982)
2. Koch, C., Ullman, S.: Shifts in selective visual-attention towards the underlying neural circuitry. Hum. Neurobiol. 4, 219–227 (1985)
3. Itti, L., Koch, C.: Feature combination strategies for saliency-based visual attention systems. J. Electron Imaging 10(1), 161–169 (2001)
4. Mallet, S.G.: A theory for multiresolution signal decomposition: The wavelet representation. IEEE Tran. on PAMI 11, 674–693 (1989)
5. Hyvarinen, A., Hoyer, P.O.: Emergence of phase and shift invariant features by decomposition of natural images into independent feature subspace. Neural Computation 12(7), 1705–1720 (2000)
6. Hubel, D.H., Wiesel, T.N.: Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J. Physiol. 160, 106–154 (1962)
7. Blakemore, C., Cooper, G.F.: Development of the brain depends on the visual environment. Nature 228, 477–478 (1970)
8. von der Malsburg, C.: Self-organization of orientation sensitive cells in the striate cortex. Kybernetik 14, 85–100 (1973)
9. Willshaw, D.J., von der Malsburg, C.: How patterned neural connections can be set up by self-organization. Proc. R. Soc. Land. B. 194, 431–445 (1976)
10. Kohonen, T.: Self-organized formation of topographically corerct feature maps. Biol. Cybern. 43, 59–69 (1982)
11. Horn, B.K., Schunck, B.G.: Determining optical flow. Artif. Intell. 17, 185–203 (1981)

Intelligent Face Image Retrieval Using Eigenpaxels and Learning Similarity Metrics

Paul Conilione and Dianhui Wang

Department of Computer Science and Computer Engineering
La Trobe University, Melbourne, VIC 3086, Australia
dh.wang@latrobe.edu.au

Abstract. Content-based Image Retrieval (CBIR) systems have been rapidly developing over the years, both in labs and in real world applications. Face Image Retrieval (FIR) is a specialised CBIR system where a user submits a query (image of a face) to the FIR system which searches and retrieves the most visually similar face images from a database. In this paper, we use a neural-network based similarity measure and compare the retrieval performance to Lp-norm similarity measures. Further we examined the effect of user relevance-feedback on retrieval performance. It was found that the neural-similarity measure provided significant performance gains over Lp-norm similarity measures for both the training and test data sets.

1 Introduction

Content-based image retrieval (CBIR) aims to retrieve images from a database that match a users query based on image content. The past decade has seen rapid developments in CBIR research. With many techniques being developed to solve problems in medical imaging, art, culture, and web-based image retrieval [1]. In the special case of Face Image Retrieval (FIR), a user wishes to find similar faces to a mental image of a person they have seen or imagined. The user can submit a query in the form of an image (query-by-example), sketch or description. FIR can be used in law enforcement agencies needing to search large databases of faces for a suspect, casting agencies for finding suitable looking cast, or for an individual to search their own photo albums of friends and family.

The first step in FIR systems is to extract features from the images that are low dimensional and capture the visual content of the images. However, this is challenging due to the variations in appearance of faces in images. A face can appear at different scales, head poses, facial expressions, occlusions (such as eyewear or facial hair), background clutter, illumination variation and rotation of the face along the optical axis of the camera. To simplify the problem, researchers have used "well-framed" frontal images of faces. Where "well-framed" means that the object of interest must have small variations in its location, orientation and size in each image [2]. One of the most common feature extraction method is the eigenface method where principle component analysis (PCA) is applied to well-framed images. The resulting eigenvectors are used to transform the images into a lower dimensional space where they are compared using a similarity measure (e.g. Euclidean) [3]. Eigenface-based methods have been enhanced by using Linear

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 792–799, 2009.

Discriminant Analysis, which considers class information to help distinguish the faces of different people in the feature space [4]. The recently developed eigenpaxel method transforms small windows of pixels to a lower dimensional space by apply PCA locally rather than on the whole image as is the case for the Eigenface method. The eigenpaxel method is analogues to how the local receptive fields in the visual system of mammals function [5].

Relevance feedback (RF) [6] aims to refine the retrieved images according to the user indicating to the system which images are relevant or not. Even if automatic extraction of high level concepts is possible, it's impossible to capture all concepts that will satisfy all users over time as each users wishes are different and can vary over time. Hence, RF has the advantage of being able to adapt to each user and situation.

The use of Neural Network (NN) based RF in CBIR has been widely used by researchers [7,8], whilst very little research has been conducted in FIR. The goal of this paper is to present a neural network based Intelligent FIR (IFIR) system using RF that accepts images as the query (query-by-example). We examine the recognition power of the eigenpaxel feature extraction method and explore the effect of RF on performance. Section 2 presents a detailed description of our IFIR system. Section 3 presents the results of the proposed system and discussion. Section 4 concludes the paper.

2 System Description

2.1 Overview

Figure 1 presents the overall design of the IFIR system presented in this paper.

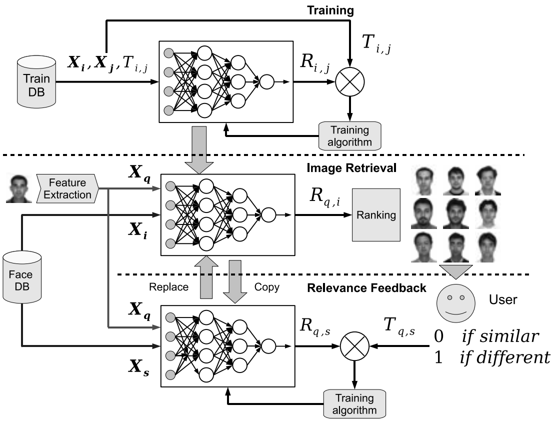


Fig. 1. Overall design of the IFIR system, with off-line neural network training. On-line feature extraction and image retrieval. Finally there is neural network re-training with RF from the user.

The system has three components, the first is the training of the neuro-similarity measure NN using pairs of image features $\mathbf{X}_i, \mathbf{X}_j$ with associated target values $T_{i,j}$ (not in diagram). The second is image retrieval where feature extraction is performed on the query image, yielding $\mathbf{X}_q$, which is compared to feature vectors of the i-th image in the database $\mathbf{X}_i$ using the trained NN. The resulting output from the network $R_{q,i}$ is a measure of relevance and is used to rank the images, which are then displayed to the user. For RF, the user can select images as being "similar" or "different" and this information is used to re-train the NN. The updated NN replaces the old NN and the similarity between the query image and the database images are recalculated, with the newly ranked images displayed to the user.

2.2 Neural Similarity

The NN can be used as a similarity function where the input is the combined feature vectors of the query and database images $(\mathbf{X}_q, \mathbf{X}_i)$ and the output is the relevance $R_{q,i}$ between the two images. The first stage in the IFIR system is the off-line training of the neural network. This is achieved by presenting a set of image feature vector pairs $(\mathbf{X}_i, \mathbf{X}_j)$ with associated degree of similarity $T_{i,j}$. Where T is assigned by the designer for off-line training.

2.3 User's Relevance Feedback

Once the images are ranked by relevance, the user marks a number of retrieved database images as "similar" or "different". Each selected image, s, is assigned a target value for a given query image q by $T_{q,s} = 0$ if "similar" and $T_{q,s} = 0$ if "different". Where s is the indices of database images selected by the user. The neural network is retrained with the feature vectors from the query/database image pairs $(\mathbf{X}_q, \mathbf{X}_s)$ and associated target values $T_{q,s}$. The original NN is replaced with the updated NN and the database images are re-ranked. RF process can be repeated until the user has found the face(s) they are seeking.

2.4 Feature Extraction

A 2D image can be viewed as having N-dimensional point in the image space ($N = width \times height$). The aim of the Eigenpaxel feature extraction is to map these points to a lower dimensional space, using neighbouring pixel information for the transformation [5]. From Alg. 1 the Eigenpaxels $\mathbf{U}^{\mathrm{D}}$ with ρ principle components are derived using Alg. 2, where D is the training set of d images ($w \times h$ pixels) and N is the number of paxels randomly selected from each image. Then the images are eigenfiltered using $\mathbf{U}^{\mathrm{D}}$ with $s\%$ overlapping windows and averaged with non-overlapping window α, Alg. 3. Finally, PCA is then applied to the eigenfiltered and averaged training images to yield the eigenvectors $\mathbf{V}^{\mathrm{D}}$ for the top ρ principle components for feature vector compression.

Feature extraction for database or query images is performed by applying eigenpaxel filtering, Alg. 3 using $\mathbf{U}^{\mathrm{D}}$ and then apply PCA to $\mathbf{S}$ with $\mathbf{V}^{\mathrm{D}}$ for data compression to yield the final feature vector $\mathbf{X}$.

2.5 Training and Performance Evaluation

Face Image Data Set. The AR data set [9] used in this work consists of photographs of people taken over two sessions, two weeks apart. The first session (session 1) had 135

Algorithm 1. Eigenpaxel creation and Feature Vector Compression.

```
procedure CREATEEIGENPAXELEIGENVECTOR($D, N, p, q, \rho, s, \alpha, \tau$ )
    $\mathbf{U}^{\mathrm{D}} = \{$CALCULATEEIGENPAXELS($D_i, p, q, N, \rho$)$\}|_{1,\ldots,d}$
    $\mathbf{S}^{\mathrm{D}} = \{$EIGENPAXELFILTER($D_i, \mathbf{U}^{\mathrm{D}}, s, \alpha$)$\}|_{1,\ldots,d}$
    $\mathbf{V} = \{\mathbf{V}_i\}|_{i=1,\ldots,\tau}$ set of eigenvectors from apply PCA to set of paxels $\mathbf{S}^{\mathrm{D}}$
    return $\mathbf{U}^{\mathrm{D}}, \mathbf{V}^{\mathrm{D}}$
end procedure
```

Algorithm 2. Eigenpaxel algorithm [5].

```
procedure CALCULATEEIGENPAXELS($D, N, p, q, \rho$)
    $\mathbf{P} = [\mathbf{P}_1, \ldots, \mathbf{P}_{(N \times d)}] \leftarrow$ randomly selected paxels from $D$.
    $[P_{i,j}]_{p \times q} \rightarrow [P_k]_{1 \times (p \times q)}$ is reformatted into a column vector.
    $\mathbf{U} = \{\mathbf{U}_i\}$, $i = 1, ..., \rho$ set of eigenvectors from apply PCA to set of paxels $\mathbf{P}$
    return $\mathbf{U}$
end procedure
```

people, then two weeks later, 120 people from first session were photographed again (session 2). There were no restriction on individuals to keep their hairstyle, cloths or makeup the same between sessions.

During both session 1 and session 2, each person was photographed 13 times with different expressions, lighting conditions, and with and without sunglasses or scarf. The original images were 768 x 576 in 24 bit colour, but were cropped and rescaled to 101 x 120, 8 bit gray scale images for this paper.

Feature Extraction Parameters. Using 16 x 16 paxels, with $N = 50$ randomly selected paxels per image, the eigenpaxels $\mathbf{U}^{\mathrm{D}}$ with $\rho = 20$ principle components, were derived using the method in Alg. 2. The training set, D, composed of 135 images of 135 people with neutral expression from session 1 of the AR database. The eigenvector for feature vector compression $\mathbf{V}^{\mathrm{D}}$ with $\tau = 100$ principle components was derived from the Eigenfiltered images using a sliding window with $s = 75\%$ overlap and averaging window size of $\alpha = 2$. The features where then extracted from all images from session 1 and session 2 data sets using $\mathbf{U}^{\mathrm{D}}$ eigenpaxels and $\mathbf{V}^{\mathrm{D}}$.

Neural Network Parameters. The feedforward neural network consisted of a fully connected three layer network with a single output neuron and all neurons used the logarithmic sigmoid activation function. The number of hidden neurons in the single

Algorithm 3. Eigenpaxel filtering and Averaging algorithm [5].

```
procedure EIGENPAXELFILTER($I, \mathbf{U}, p, q, s\ \alpha$)
    $P = \{\mathbf{P}_{i,j}\}$, paxels from $s\%$ overlapping sliding window at positions $i, j$ from $I$.
    $\mathbf{Q} = \{\mathbf{Q}^k\}|_{k=1,\ldots,\rho}$, where $Q^k_{i,j} = \mathbf{U}^T_k \mathbf{P}_{i,j}$, and $\mathbf{U}_k$ is the k-th eigenpaxel.
    $\mathbf{S} = \{\mathbf{S}_k\}|_{k=1,\ldots,\rho}$ is the set of averaged $\mathbf{Q}^k$, where $S^{I,k} = \frac{1}{\alpha^2} \sum_{i,j}^{i+\alpha,j+\alpha} |\mathbf{Q}^{I,k}_{i,j}|$
    return $\mathbf{S}$
end procedure
```

hidden layer was varied between 30 to 150, and the performance was measured using average mean-rank (1) and overall precision (2). The network weights were initialised using the Nguyen and Windrows algorithm. For initial off-line training the resilient-backpropagation algorithm was used with 80% of the training data used for training and 20% used for validation. Once the validation mean-square-error reached 0.001, training was stopped.

Off-line Training Data Set. For off-line training of the neuro-similarity measure we created a training set from session 1 images. Which consisted of image feature-vector pairs with an associated target (similarity) value, $[\mathbf{X}_i\mathbf{X}_j], T_{i,j}$. As each feature-pair has 200 elements the neural network had 200 input nodes. Two types of pairs were generated, intra-pairs which are from the same person, and inter-pairs that are from different people. For images i, j that belong to the same person (intra-pairs), the target value $T_{i,j}$ was randomly assigned a value in the range $[0.0, 0.1]$. If images i, j are of two different people (inter-pairs) then $T_{i,j}$ was randomly assigned a value in the range $[0.9, 1.0]$. A total of ≈16,000 intra-pairs were created from session 1 as examples of similar images for training. Whilst approximately 62,000 inter-pairs were generated from session 1 as examples of dissimilar images. The intra-class and inter-class pairs were combined to produce ≈78,000 pairs for training the neural network.

Neural Network Training for RF. The neural-similarity with RF allows the user to label images over 5 feedback loops. The neural network was updated using the gradient decent with moment training algorithm and was trained for 300 epochs with a learning rate of 0.001 and momentum term of 0.9. We used a network with 50 hidden neurons, as it was found to have the best mean-rank of the topologies tested. The effect of the number of feedback images per RF session on retrieval performance was explored by allowing the user to select 1 to 5 true-positive (TP) images (relevant images labelled as "similar"), 1 to 5 true-negative (TN) images (irrelevant images labelled as "different"), or 1 to 5 each of TP and TN examples over the course of a single RF session.

Retrieval Performance and Comparison to L_p-norm. Performance evaluation was done by comparing each image in a query set to all database images using trained neural network. The query and database sets were created from the session 1 and session 2 images of the AR database. From session 1 the query set had 135 images of 135 people with neutral expression and the remaining 1620 images were used for the database (denoted DB1-Q1). Query set from session 2 had 120 images of 120 people with neutral expression and the remaining 1200 images were used for the database (denoted DB2-Q2).

We compared the retrieval performance of the IFIR and L1-norm, L2-norm (Euclidean) and $L\infty$-norm distance-based similarity measures. The Lp-norm distances are defined as $d(X_q, X_i)_{\mathtt{L1}} = \sum_{k=1}^{100} |X_k^i - X_k^q|$, $d(X_q, X_i)_{\mathtt{L2}} = \sqrt{\sum_{k=1}^{100} \left(X_k^i - X_k^q\right)^2}$ and $d(X_q, X_i)_{\mathtt{L}\infty} = max(|X_1^i - X_1^q|, ..., |X_{100}^i - X_{100}^q|)$ where q is the query image, i is the i-th database image and k is the k-th feature.

Performance Metrics. The performance was measured by using the average mean-rank (1) and overall precision (2). Where μ_q is the mean-rank for a given query q, and $r_{q,i}$ is the rank of relevant database image i in the returned results. μ_q is sensitive to the ranking order of the relevant images. The higher the rank of relevant images from the database, the closer μ_q is to 1. Whilst the lower the rankings of relevant retrieved images, μ_q approaches 0.

$$\mu = \sum_{q=1}^{Q} \mu_q \text{ where } \mu_q = \frac{N_q\left(N_q + 1\right)}{2\sum_{i=1}^{N_q} r_{q,i}} \tag{1}$$

The overall precision is the percentile of database images that are relevant with respect to the query image in the first 12 retrieved images. Where p_q is the precision for a query image q, N_q is the number of relevant images in the database for query image q and is equal to 12.

$$p = \frac{1}{Q}\sum_{q=1}^{Q} p_q \text{ where } p_q = \frac{1}{12}\left(\sum_{i=1, r_{q,i} \leq 12, T_{q,i}=0}^{1620} 1\right). \tag{2}$$

3 Results and Discussion

3.1 Training Performance

Table 1 shows the resulting average mean-rank and overall precision for networks of varying topology for DB1-Q1 and DB2-Q2. The best overall results were achieved using networks with 50 neurons in the hidden layer. There was a significant drop in performance for networks with over 100 neurons in the hidden layer.

Table 1. Retrieval performance of trained neural networks using DB1-Q1 and DB2-Q2 respectively

Data set	**DB1-Q1**		**DB2-Q2**	
Performance	μ	p	μ	p
30	0.9555	0.9614	0.2390	0.4755
50	0.9568	0.9583	0.2485	0.5161
70	0.9399	0.9447	0.2305	0.4195
100	0.9518	0.9534	0.2845	0.5168
120	0.9280	0.9328	0.1724	0.3536
150	0.8846	0.8937	0.1732	0.3508

Clearly the images from session 1 (DB1-Q1) have significantly better results than those from session 2 (DB2-Q2). This is expected given session 1 images were originally used to create the data set of image-pairs for network training and the images in Q1 were used to derive the eigenpaxels $\mathbf{U}$ and eigenvector $\mathbf{V}$.

Table 2. Comparison between neuro-similarity and Lp-norm similarity results using DB1-Q1 and DB2-Q2

Data set	**DB1-Q1**		**DB2-Q2**	
Performance	μ	p	μ	p
NN-50	0.9568	0.9583	0.2485	0.5161
L_1-norm	0.1306	0.4728	0.0802	0.4262
L_2-norm	0.1241	0.4355	0.0879	0.4208
L_∞-norm	0.0700	0.3486	0.0589	0.3565

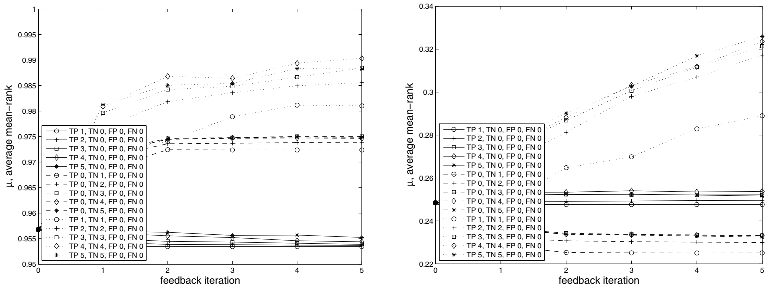
3.2 Comparison of Lp-Norm and Neural-Based Retrieval

Table 2 compares the performance of the NN with 50 neurons in the hidden layer, which had the best retrieval results of network topologies and the Lp-norms for DB1-Q1 and DB2-Q2.

From Tab. 2 the neural similarity retrieval provides better average mean-rank and precision compared to the Lp-norm methods for both data sets. Given the retrieval performances of DB2-Q2 is significantly lower compared to DB1-Q1 both neural-similarity and Lp-norm similarity measures. It would imply that the generalisation ability of eigenpaxel feature extraction method is poor, rather than a problem with the similarity measure itself.

3.3 Relevance Feedback Performance

Using the NN with 50 neurons in the hidden layer for RF experiments, Fig. 2(a) and Fig. 2(b) plots the average mean-rank results of RF over 5 iterations using the DB1-Q1 and DB2-Q2 data sets respectively. The number of TP and TN examples can be grouped into three types of feedback. Those with 1 to 5 TP examples (solid line), 1 to 5 TN examples (dashed line) and 1 to 5 each of TP and TN examples (dotted line).



(a) DB1-Q1.

(b) DB2-Q2.

Fig. 2. Feedback performance of IFIR system with μ vs number of feedback loops

For both DB1-Q1 and DB2-Q2 data sets, it is clear that having a user select both TP and TN examples for re-training the NN provides significant improvement in retrieval performance compared to feedback using only TP or TN examples.

4 Conclusions

We tested the eigenpaxel based feature extraction method for our neural-similarity based Intelligent Face Image Retrieval (IFIR) system. We found that the eigenpaxel based feature extraction method generalised poorly for face images that were not used for calculating the original eigenpaxels or eigenvectors for feature compression. However, the neural-similarity measure had significantly better retrieval results compared to L1-norm, L2-norm and $L\infty$-norm for both training and test data sets. The neural-similarity with relevance feedback was found to be very effective at improving retrieval performance as measured with mean-rank and precision. However, this was only the case when the user provides both true positive (TP, relevant image labelled as "similar") and true-negative (TN, irrelevant image labelled "different") examples for feedback. Otherwise selecting either TP or TN examples only during a feedback session resulted in only a small improvement, or even a drop in performance.

References

1. Datta, R., Joshi, D., Li, J., Wang, J.Z.: Image retrieval: Ideas, influences, and trends of the new age. ACM Comput. Surv. 40(2), 1–60 (2008)
2. Lai, P.J., Wang, J.H.: Facial Image Database for Law Enforcement Application: an Implementation. In: 37th IEEE International Carnahan Conference on Security Technology, pp. 285–289 (2003)
3. Turk, M., Pentland, A.: Eigenfaces for Recognition. J. Cog. Neurosci. 3(1), 71–86 (1991)
4. Swets, D., Weng, J.: Using Discriminant Eigenfeatures for Image Retrieval. IEEE Trans. Pattern Anal. Mach. Intell. 18(8), 831–836 (1996)
5. McGuire, P., D'Eleuterio, G.: Eigenpaxels and a Neural-network Approach to Image Classification. IEEE Trans. on Neural Networks 12(3), 625–635 (2001)
6. Zhou, X.S., Huang, T.S.: Relevance Feedback in Image Retrieval: A Comprehensive Review. Multimedia Syst. 8(6), 536–544 (2003)
7. Wang, D.H., Ma, X.H.: A hybrid image retrieval system with user's relevance feedback using neurocomputing. Informatica - An International Journal of Computing and Informatics 29, 271–279 (2005)
8. Wang, B., Zhang, X., Li, N.: Relevance Feedback Technique for Content-based Image Retrieval using Neural Network Learning. In: IEEE International Conference on Machine Learning and Cybernetics, pp. 3692–3696 (2006)
9. Martinez, A.M., Benavente, R.: The AR Face Database. Technical Report 24, Computer Vision Center (1998)

The Role of the Infant Vision System in 3D Object Recognition

Roberto A. Vázquez, Humberto Sossa, and Beatriz A. Garro

Centro de Investigación en Computación – IPN
Av. Juan de Dios Batiz, esquina con Miguel de Othon de Mendizábal
Ciudad de México, 07738, México
ravem@ipn.mx, hsossa@cic.ipn.mx, bgarrol@ipn.mx

Abstract. Recently, it was shown how some metaphors, adopted from the infant vision system, were useful for face recognition. In this paper we adopt those biological hypotheses and apply them to the 3D object recognition problem. As the infant vision responds to low frequencies of the signal, a low-filter is used to remove high frequency components from the image. Then we detect subtle features in the image by means of a random feature selection detector. At last, a dynamic associative memory (DAM) is fed with this information for training and recognition. To test the accuracy of the proposal we use the Columbia Object Image Library (COIL 100).

1 Introduction

Object recognition is one of the most researched areas in computer vision. Most of the reported methods can be categorized as geometric-based or appearance-based. Appearance-based methods have attracted much attention in the last years. They learn a model of the object's appearance in a two-dimensional image under different poses and illumination conditions.

Several appearance-based methods have been proposed to recognize 3D objects. In [2], the authors show that 3D objects can be recognized from the raw intensity values in 2D images (*pixel-based representation*). In [3] and [4], Murase and Nayar developed a parametric eigenspace method to recognize 3D objects directly from their appearance. A support vector machine is applied in [6] to recognize 3D objects from 2D images. In [6] SVMs are also used for 3D object recognition. Experiments are presented a subset of the COIL-100 data set [8]. Recently, in [14] the authors proposed a new technique where they give a partial solution to the 3D object recognition problem (subset of COIL-100 from 0 to 100 degrees) combining some aspects of the infant vision system with associative memories.

It is well known that during early developmental stages, there are communication pathways between the visual and other sensory areas of the cortex, showing how the biological network is self-organizing. It has been hypothesized that the functional role of perception is to capture the statistical structures of the sensory stimuli such that corresponding actions could be taken to maximize the chances of survival (see [10] for details). Barlow hypothesized that for a neural system, one possible way of capturing the statistical structure was to remove the redundancy in the sensory outputs [10].

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 800–807, 2009.

Taking into account these theories and adopting the ideas described in [13], we extend the appearance-based method, proposed in [14], for giving a full solution (from 0 to 360 degrees) to 3D object recognition problem based on some biological aspects of infant vision [8] and [11]. A dynamic associative memory (DAM) is used as a learning device to recognize different objects from different points of view. Due to the infant vision responds to low frequency signals, a low-filter is first used to remove high frequency components from the image. Then we detect subtle features in the image by means of a random selection of stimulating points. At last, the DAM is fed with this information for training and recognition. To test the accuracy of the proposal, we use the Columbia Object Image Library (COIL 100).

The DAM can be seen as a particular kind of neural network specially designed to recall output patterns in terms of input patterns that might appear altered by some kind of noise. This model is not an iterative model as Hopfield's model. The principal difference of this model against other classic models is that once trained, during recalling phase the synapses' values could change as a respond to an input stimulus. The formal set of propositions that support the correct functioning of this model and the main advantages against other classical models can be found in [15].

2 The Infant Vision System

Categorization is a fundamental information processing capability that allows reliable recognition and response to novel examples of familiar category members. Categorization of objects enables people to allocate cognitive resources efficiently. Because of the adaptive nature of categorization, it is not surprising that categorization abilities emerge early in infancy. Children have several capabilities such as learning, memory and recognition. Children emerge from the infancy period with the ability to form and retain explicit memories of past events, recent research has made clear that even very young infants have the capability to retain information over long delays [9].

Scientists used to believe that the newborn's brain was just a smaller version of the adult's brain, which was completely "wired" at birth. Today we know that the baby's brain is a dynamic structure; it makes many new connections each day as it grows. Researchers have established that infants appear to recognize familiar stimuli early in the first year of life (e. g. a mother's face). There are several evidences that indicate the infants are capable to efficiently perform two of the most challenging and complex problems in pattern recognition: face and 3D object recognition [8].

Vision is limited at birth (Fig. 1), such that only low spatial frequencies are processed [1]. The Linear System Model (LSM) proposed by Banks and Salapatek, is based on the assumption that newborns prefer to look at what they better see.

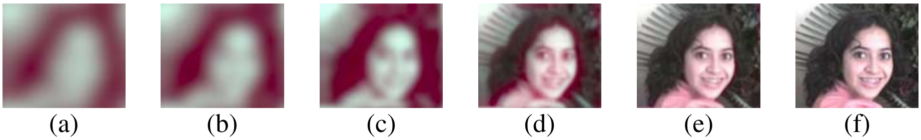
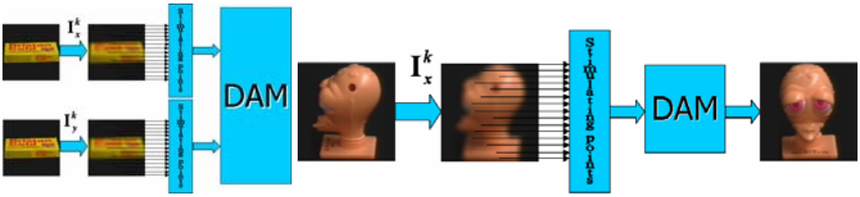


Fig. 1. Images perceived by an infant. (a) Newborn. (b) 8-week old. (c) 16-week old. (d) 3-month old. (e) 6-month old. (f) Adult.

3 Description of the Proposal

The proposal consists of a DAM used to recognize different 3D objects; some interesting applications of this model are described in [12], and [13]. As the infant vision responds to low frequency signals, a low-filter was first used to remove high frequency components from the image. We proceeded to detect subtle features in the image by means of a random selection of stimulating points. At last, the DAM was fed with this information for training and recognition, see Fig. 2.



(a) (b)

Fig. 2. A general schema of the proposal. (a) Building phase. (b) Recall phase.

It is important to mention that instead of using a filter that exactly simulates the infant vision system behavior at any stage we use a low-pass filter (average filter) to remove high frequency. As equal as the infant vision system, this filter eliminates high frequency components from the pattern.

To simulate the hypothetical idea of how an infant detects subtle features, some pixels (stimulating points) of pattern $\mathbf{x}^k$ are random selected, where k defines the class of the pattern. These stimulating points **SP** are used by the DAM to determine an active region and are given by $\mathbf{sp}\in\left\{\square^{+}\right\}^{c}$ where c is the number of used **SP**. $sp_i = random(n), i = 1,\ldots,c$ where n is the size of the pattern. To determine the active region, the DAM stores during training phase an alternative simplified version of each pattern $\mathbf{x}^k$ given by:

$$\mathbf{ss}^k = ss\left(\mathbf{x}^k\right) = \mathbf{x}^k\Big|_{\mathbf{sp}} = \left\{x_{sp_1}^k,\ldots,x_{sp_c}^k\right\} \tag{1}$$

During recalling, each element of an input simplified pattern $\tilde{\mathbf{x}}^k\Big|_{\mathbf{sp}}$ excites some of these regions and the most excited region will be the active region. To determine which region is excited by an input pattern we use:

$$b = \arg\min_{k=1}^{p}\left|\left[ss\left(\mathbf{x}\right)\right]_i - \mathbf{ss}_i^k\right| \tag{2}$$

For each element of $\tilde{\mathbf{x}}^k\Big|_{\mathbf{sp}}$ we apply eq. 2 and the most excited region (the region that more times was obtained) will be the active region.

We supposed that the most relevant information that best describes an object in an image is concentrated in its center. In general, when humans pay attention to a particular object, most of the time humans they focus their sight in the center of the field vision. Trying to simulate this, we use a Gaussian random number generator.

Building of the DAM is done as follows:

```
Let $\mathbf{I}_x^k$ and $\mathbf{I}_y^k$ an association of images and $c$ be the number of stimulating points.
1. Take at random a stimulating point $sp_i, i = 1,\ldots,c$ .
2. For each association:
    a. Select filter size and apply it to the stimulating points in the images.
    b. Transform the images into a vector ($\mathbf{x}^k$, $\mathbf{y}^k$) by means of the standard image scan method.
3. Train the DAM and compute the simplified patterns by using eq. 1.
```

Pattern $\mathbf{I}_y^k$ can be recalled by using its corresponding key image $\mathbf{I}_x^k$ or distorted version $\tilde{\mathbf{I}}_x^k$ as follows:

```
1. Use the same stimulating point, $sp_i, i = 1,\ldots,c$ and filter size as in building phase.
2. Apply filter to the stimulating points in the images.
3. Transform the images into a vector by means of the standard image scan method
4. Determine active region using equation 2.
5. Recall the output image as described in recalling procedure [15].
```

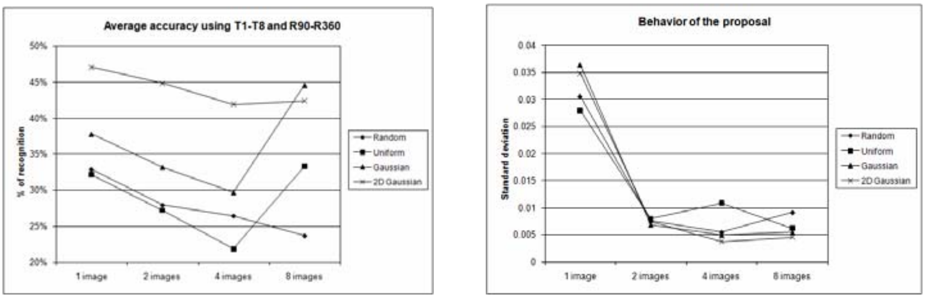
4 Experimental Results

To test the accuracy of the proposal, we have used the COIL 100 database composed by 100 objects (72 images per object). We have used different subsets of COIL 100 for training (T1, T2, T4 and T8) and recognition (R90, R180, R270 and R360). T1 is composed by 1 image of each object at 0°. T2 is composed by 2 images of each object at 0° and 180°. T4 is composed by 4 images of each object at 0°, 90°, 180° and 270°. T8 is composed by 8 images of each object at 0°, 45°, 90°, 135°, 180°, 225°, 270° and 315°. R90 is composed by 18 images of each object from 0°-85°. R180 is composed by 36 images of each object from 0°-175°. R270 is composed by 54 images of each object from 0°-265°. R360 is composed by 72 images of each object from 0°-355°.

Before training the DAM, each image was transformed into an image pattern. For this, each bmp file was read from left to right and up and down; each RGB pixel (hexadecimal value) was transformed into a decimal value. Finally the information was stored into an array. The DAM was trained in the auto-associative way using the building procedure described in section 3. Once trained the DAM we proceeded to test the proposal with three sets of experiments. We have trained the DAM using each subset for training and we have tested the accuracy of the proposal using each subset for recognition (2400 experiments were performed divided in three sets). At last, we tested the accuracy using three different numbers of **SPs**.

First set of experiments. In this set of experiments we compared the accuracy of the proposal using four different random number generators. The first generator generates uniformly distributed random numbers (uniform 1) in intervals. The second generator also generates uniformly distributed random numbers (uniform 2). The third and forth ones generate Gaussian random numbers based on the polar form of the Box-Muller transformation. For the third generator, we generated random numbers over the image transformed into a vector using a mean of 8191.5 and a standard deviation of 4729.6. For the forth generator we generated random numbers over axis *x* and *y* of the image by using a mean of 63.5 and a standard deviation of 37.09. By using this generator we tried to approximate the way humans focus their sight to the center of the field vision.

Discussion 1. In average, the accuracy of the proposal when using the first two generators was of 27%. For Gaussian generators, the accuracy of the proposal was of 36% and 44% respectively. Despite of the accuracy obtained, we observed that the results were better using Gaussian random numbers over axis *x* and *y* on an image than other generators. In this experiments we tested only with 1000, 2000 and 3000 stimulating points and accuracy obtained converged almost to the same results.



(a) (b)

Fig. 3. (a) Average accuracy of the proposal using different random selection techniques. (b) Behavior of the proposal using the different sets for training and recalling.

On the other hand, we supposed that if we increased the number of images during training the accuracy of the proposal could be increased. However, the results obtained indicated that if the number of images is increased during training, the accuracy diminishes; refer to Fig. 3 (a). Furthermore, the accuracy for R90-R360 when using T1 was so different, high accuracy for R90 and low accuracy for R360. This is a normal result because we could not expect recognizing something that we would never have seen before. However, if T2-T8 is used, the accuracy of the proposal is almost the same for R90-R360; see Fig. 3(b).

Second set of experiments. In this set of experiments we modified the value of the standard deviation of generator Gaussian 2D. When using this generator we increased the accuracy of the proposal and we also approximated how a person focuses his sight at the center of the field vision. When humans focus their sight, they also perceive information from the periphery to the center field vision. We could control the radius of the center to the periphery by adjusting the value of standard deviation.

Discussion 2. In the previous set of experiments a standard deviation of 37.09 was used, this means that SPs where generated from the whole image. The reduction of the standard deviation caused that SPs get concentrated in the center of the image. In Fig. 4 we show the accuracy of the proposal when varying the value of standard deviation (due to space limitations we only report the results using R360). SD-x is the value of the standard deviation 37.09 minus the value of x. As can be appreciated from this figure, when the standard deviation is reduced, the accuracy of the proposal increases, but when the stimulating points are too concentrated in the center of the image the accuracy of the proposal tends to decrease. This result could well describe the behaviour of humans when they watch an object. For example, if the person is far away from an object, then the person could see the object but it could be difficult to recognize it due to some distractions (noise); if the person gets closer (diminishing the standard deviation) the person could see and recognize the object without distractions; but if the person gets so closer (diminishing so much the standard deviation) the person could see the object but it could be difficult to recognize it due to he needs more information of the object. The two best accuracies, changing the value of the standard deviations, were of 94% and 96% when using SD-20 with T4 and T8 respectively.

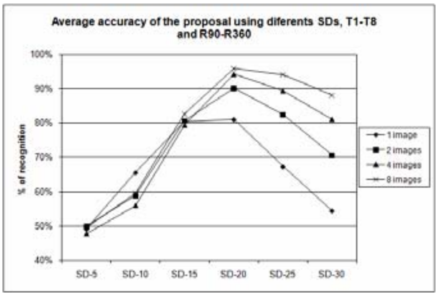
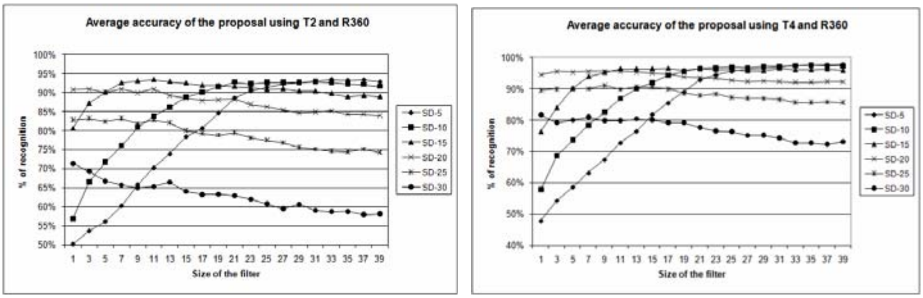


Fig. 4. Accuracy of the proposal using different standard deviations and different sets for training and recognition

Third set of experiments. In this set of experiments we removed the high frequencies of the images in order to improve the accuracy of the proposal. By removing high frequencies, as we will see, unnecessary information is eliminated; with the help of the DAM efficient object recognition can be attained. Although this is not a plausible solution to model the infant vision system, as we previously said, we have used an average filter. In this set of experiments we applied the filter to the stimulating point in the images. We tested different sizes of the filter (1-39) combine with different SD-x.

Discussion 3. As can be appreciated from Fig. 5(a) and Fig 5(b), the accuracy of the proposal when using SD-5, SD-10 and SD-15 increases when the size of the filter is increased. For SD-20, SD-25 and SD-30 the accuracy of the proposal slightly decreases when the size of the filter increases. The two best accuracies obtained with T2 were of 93% and 92% using SD-5 and SD-10 respectively. The two best accuracies obtained with T4 were of 98% and 97% using SD-5 and SD-10 respectively.

In average, by removing high frequencies and by selecting at random stimulating points, and by using a Gaussian number generator over axis x and y, contributes to eliminating unnecessary information; with the help the DAM efficient object recognition can



(a) (b)

Fig. 5. Accuracy of the proposal simulating the behavior of the infant vision system. (a) Behavior obtained using T2 and different SDs. (b) Behavior obtained using T4 and different SDs.

be obtained. In general, the accuracy of the proposal surpasses the 90% of recognition and with some configurations we up-performed this result until 99%.

The results obtained with the proposal through several experiments were comparable with those obtained by means of a PCA-based method (99%). Although PCA is a powerful technique it consumes a lot of time to reduce the dimensionality of the data. Our proposal, because of its simplicity in operations, is not a computationally expensive technique and the results obtained are comparable to those provided by PCA. An analysis of the computational complexity of random selection requires only $O(rn)$ whereas PCA requires $O(p2n)+O(p3)$.

5 Conclusions

In this paper we have shown that by applying some aspects of the infant vision system it is possible to enhance the performance of an associative memory. Also we show that is possible its application to complex problems such as 3D object recognition.

The biological hypotheses of this method are based on the role of the response to low frequencies at early stages, and some conjectures concerning how an infant detects subtle features (stimulating points) in objects. We used a DAM to recognize different images of a 3D object. As the infant vision responds to low frequency signals, a low-filter is first used to remove high frequency components from the image. Then we detect subtle features in the image by means of a random selection of stimulating points. At last, the DAM is fed with this information for training and recognition.

Through several experiments (using different training sets and simulating the behavior of the infant vision system) we have shown how the accuracy of the proposal can be increased by using a Gaussian number generator over axis x and y on an image. The way as we humans focus our sight to the center of the field vision and perceive information from the periphery to the center field vision could be simulated by adjusting the value of standard deviation.

By removing high frequencies and by randomly selecting of stimulating points contributes to eliminate unnecessary information and with the help of a DAM object recognition can be very efficient. Important to mention is that, to our knowledge, nobody has reported results of this type using an associative memory for 3D object recognition. These encouraging results suggest that the study of the behavior and neural process of the human brain could provide bio-inspired ideas to solve complex problems in pattern recognition.

The results obtained with the proposal were comparable to those obtained by means of a PCA-based method. Although PCA is a powerful technique it consumes a lot of time to reduce the dimensionality of the data. Our proposal, because of its simplicity in operations, is not a computationally expensive technique and the results obtained are comparable to those provided by PCA.

This is just the first step. Nowadays we are investigating other aspects of the infant vision system such as attention, learning and feature extraction in order to give a more general solution to this complex problem. We are sure that the study of human brain could provide us of some useful cues to solve this and other complex problems.

Acknowledgments. This work was economically supported by SIP-IPN under grant 20082948 and CONACYT under grant 46805.

References

[1] Banks, M.S., et al.: Infant Pattern Vision: a New Approach based on the Contrast Sensitivity Function. J. Exp. Child Psychol. 31, 1–45 (1981)

[2] Poggio, T., et al.: A network that learns to recognize 3d objects. Nature 343, 263–266 (1990)

[3] Murase, H.: Visual learning and recognition of 3-d objects from appearance. Int. J. Comp. Vision 14, 5–24 (1995)

[4] Nayar, S.K., et al.: Real-time 100 object recognition system. In: Proc. of IEEE Int. Conf. on Rob. and Aut., pp. 2321–2325 (1996)

[5] Nayar, S., et al.: COIL 100. Tech. Rep. No. CUCS-006-96. Dep. of Comp. Sci., Columbia University

[6] Pontil, M., Verri, A.: Support vector machines for 3d object recognition. IEEE Trans. Pattern Anal. Mach. Intell. 20(6), 637–646 (1998)

[7] Roobaert, D., et al.: View-based 3d object recognition with support vector machines. In: Proc. of IEEE Int. Workshop on NNSP, pp. 77–84 (1999)

[8] Mondloch, C.J., et al.: Face Perception During Early Infancy. Psychological Science 10(5), 419–422 (1999)

[9] Carver, L., et al.: Associations between Infant Brain Activity and Recall Memory. Devel. Sci. 3(2), 234–246 (2001)

[10] Barlow, H.: Redundancy Reduction Revisited. Network: Comput. Neural Syst. 12, 241–253 (2001)

[11] Acerra, F., et al.: Modelling aspects of face processing in early infancy. Developmental science 5(1), 98–117 (2002)

[12] Vazquez, R.A., Sossa, H.: A computational approach for modeling the infant vision system in object and face recognition. J. BMC Neurosci. 8(suppl. 2), 204 (2007)

[13] Vázquez, R.A., Sossa, H., Garro, B.A.: Low frequency response and random feature selection applied to face recognition. In: Kamel, M.S., Campilho, A. (eds.) ICIAR 2007. LNCS, vol. 4633, pp. 818–830. Springer, Heidelberg (2007)

[14] Vázquez, R.A., Sossa, H., Garro, B.A.: 3D Object Recognition Based on Low Frequency Response and Random Feature Selection. In: Gelbukh, A., Kuri Morales, Á.F. (eds.) MICAI 2007. LNCS (LNAI), vol. 4827, pp. 694–704. Springer, Heidelberg (2007)

[15] Vazquez, R.A., Sossa, H.: A new associative model with dynamical synapses. Neural Processing Letters 28(3), 189–207 (2008)

Virtual Fence for a Surveillance System

Yen San Yong, Hock Woon Hon, Yasir Salih Osman, Ching Hau Chan,
Siu Jing Then, and Sheau Wei Chau

MIMOS Berhad, Technology Park Malaysia, 57000 Kuala Lumpur, Malaysia
{yyen.san,hockwoon.hon,yassir.ali,cching.hau,tsiu.jing,
wei.sheau}@mimos.my

Abstract. This paper presents a method for defining one or more virtual restricted zones within a surveillance area which is observed with stereo cameras. When an object enters a restricted zone, the system highlights the object shown in the monitoring screen or triggers other devices to produce a visual or auditory alarm. The proposed method works by extracting the foreground objects for both the left and the right images from their respective stereo cameras. Then it estimates the object's position in terms of depth plane using image shifting and number of overlapping pixels. Finally, it determines whether there is a collision between objects and restricted zones in order to trigger an alarm where necessary. The algorithm has been tested with a series of stereo videos, in which samples of it are presented in this paper.

1 Introduction

The ever-changing climate of public safety has always been the utmost concern of any society. However, it has become increasingly difficult to maintain safety in the face of growing uncertainty and unorthodox threats. Technological advances afford us the ability to extend our sensory reach and vigilance to close the gap on security risks. Nowadays it is not enough to simply detect for security events but to identify potential risks before it manifests itself into a dangerous security event.

Intelligent visual surveillance systems [1] are currently adopted by many to counter these security risks. Even the basic task of monitoring an area is being redefined with automatic intrusion detection.

Under human visual perception, objects located nearer to a viewpoint tend to appear larger than objects located far away from that same viewpoint. However, this assumption is not always accurate, as a child standing near and an adult standing further away from a camera may look similar in size from the camera's image where in actual fact they are not. Instinctively humans are able to differentiate this but such problem presents a significant challenge for automated visual surveillance systems. Thus, machine understanding of our 3-dimensional (3D) world and the relative spaces that we have occupied are required to solve this problem.

This paper proposes a surveillance system that can automatically detect when an object enters a virtual restricted 3-dimensional zone as illustrated in Fig. 1. The proposed system employs two cameras to capture stereo images. With this system, the perspective problem is solved by deriving an object's 3D coordinates to detect intrusions and to aid automated visual surveillance systems.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 808–815, 2009.

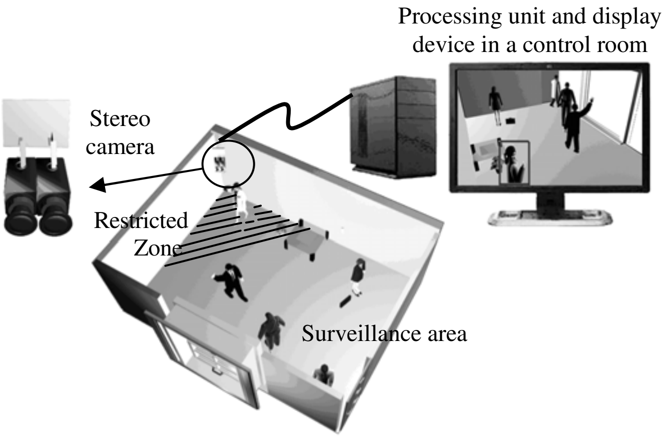


Fig. 1. Apparatus setup for virtual fence for a surveillance system

In stereo vision, solving the correspondence problem is a key point. Searching for correspondence points or features is required as the inputs for the matching process in order to calculate the parallax [2][3][4]. The processing time for this process increases when the number of objects increases or the objects having similar feature properties increases. This will affect the overall processing time in detection for a surveillance system, which is time-critical. In addition, the performance of this method depends a lot on the resultant image quality from the pre-processing stage, such as foreground extraction. For example, features such as blob area or centroid for the same object may vary between left and right images under different brightness condition. As a result, objects may not be matched or the parallax information may be wrongly calculated.

The proposed method estimates the parallax using image shifting and number of overlapping pixels. This method does not require any matching process, and thus, it reduces the processing time and the dependency on the pre-processed image quality is not that critical.

This paper is divided into 6 sections. Section 2 gives an overview on parallax whereas Section 3 demonstrates the proposed algorithm in detail. The experimental system setup is illustrated and explained in Section 4. A few experiments have been conducted using stereo surveillance video. Due to the limitation on results presentation, some video frames are extracted and presented in Section 5 as images. The paper ends with conclusion and future work as described in Section 6.

2 Parallax

Human eyes are horizontally separately and thus each eye has a slightly different view of the world. This difference in the sensed image is called binocular parallax [5][6], which is the most important depth cue for human. Stereo imaging system that mimics human eyes employs this theory to extract depth information from the left and the right (stereo) images.

Fig. 2 illustrates the concept of stereo imaging system. Object placed at different depth plane produces different separation in the stereo images. Therefore, this separation, called parallax, can be used to estimate the object's position in terms of depth plane.

The known parameters of camera setting and calibration can be applied to estimate the 3D coordinate. However, as shown in Fig. 1, the depth plane size increases as it further away from the cameras. Therefore the precision for the 3D coordinate decreases as the object placed further away from the camera.

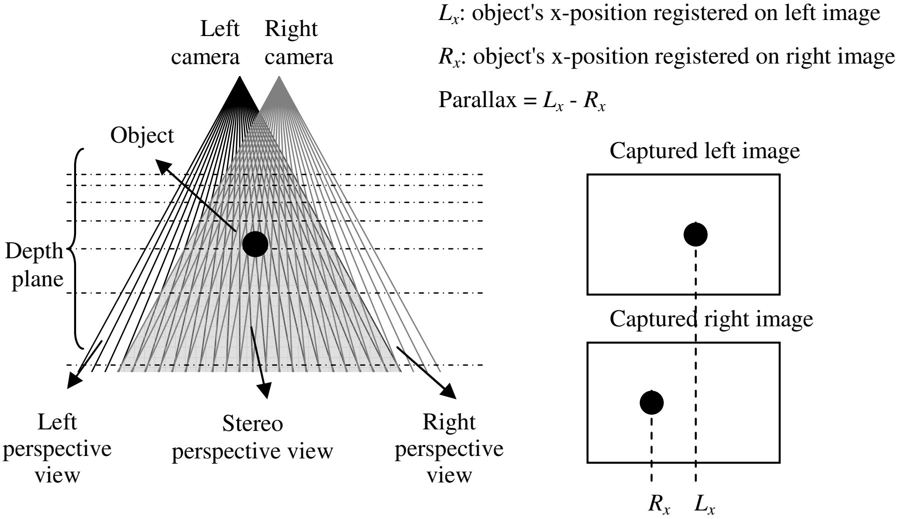


Fig. 2. Concept of stereo capturing system

3 The Proposed Algorithm

The complete algorithm for the system can be broadly divided into two sections. The pre-processing is the preparation whereas the main processing is the core of the algorithm that detects the intrusion into the restricted zone.

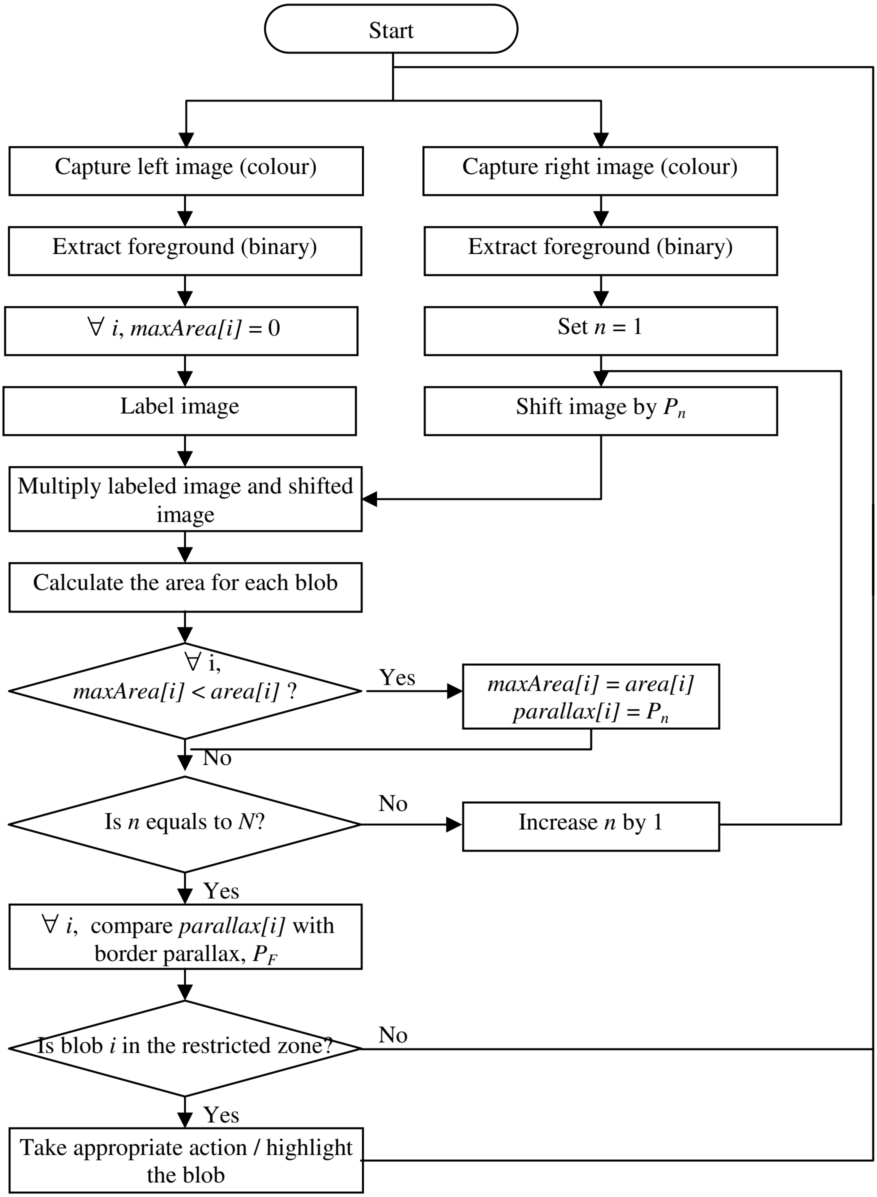
3.1 Pre-processing

There are two tasks in the pre-processing stage:

- Collect background model and
- Determine the border parallax value and the parallax list.

To perform foreground extraction, the process requires a background model. There are many methods to model a background [7].

The border parallax value, P_F, is a user defined parallax value that falls on the border of two zones. Another set of parallax values, P_n, stored in a list, is required for the process of image shifting. These parallax values, P_n, and the number of parallax values, N, are determined by the user. The more borders are defined, the longer the list.


Start
Capture left image (colour)
Capture right image (colour)
Extract foreground (binary)
Extract foreground (binary)
∀ i, maxArea[i] = 0
Set n = 1
Label image
Shift image by P_n
Multiply labeled image and shifted image
Calculate the area for each blob
∀ i, maxArea[i] < area[i] ?
Yes
maxArea[i] = area[i]
parallax[i] = P_n
No
Is n equals to N?
No
Increase n by 1
Yes
∀ i, compare parallax[i] with border parallax, P_F
Is blob i in the restricted zone?
No
Yes
Take appropriate action / highlight the blob

N = total number of parallax value in list	n = parallax counter
I = total number of labeled blob	i = blob counter
P_n = nth parallax value	*maxArea[i]* = maximum area for blob i
parallax[i] = parallax value assigned to blob i	*area[i]* = current area for blob i

Fig. 3. Flowchart for intrusion detection

3.2 Main Processing

The main algorithm, as illustrated in Fig. 3, starts with image acquisition. Stereo images are captured and processed separately.

For foreground extraction, colour image subtraction between background image and current image is performed followed by grey-scale conversion. The grey-scale conversion is adopting a simple equation as follow:

$$G(x,y) = \frac{R(x,y) + G(x,y) + B(x,y)}{3} \tag{1}$$

where $G(x,y)$, $R(x,y)$, $G(x,y)$ and $B(x,y)$, are the grey-scale, red-plane, green-plane and blue-plane for pixel (x,y) respectively.

Then, thresholding is applied to the grey-scale stereo images to produce binary foreground stereo images where the background pixels are assigned to 0 whilst the foreground pixels are assigned to 1.

The binary left image is labeled and kept as reference image whereas the right image is translated in x-axis to find the optimum parallax value for each blob. The values, P_n, listed in the parallax list are used as x-axis translation parameter. Then the labeled left image and shifted right image are multiply together to count the overlapping area between two images while maintaining the blob labeling number. If the overlapping area is greater than the area stored in *maxArea*, then the value of *maxArea* is substituted by the current area, *area*, and the current parallax shift value, P_n, is recorded in correspondence *parallax*.

After performing the shifting for every value in the parallax list, the parallax shift that having maximum overlapping pixels for the particular blob, *parallax[]*, is taken as its parallax value. This value is compared with the predefined border parallax value, P_F, to determine its zone and thus, making decision on whether an action should be taken.

4 Experiment Setup

4.1 System Setup

The proposed system consists of a pair of stereo cameras, a frame grabber with two inputs, a processor and a display device. The two cameras are placed side-by-side such that the captured stereo images look similar. The hardware setup is similar to the setup illustrated in Fig. 1.

In the experiment, the surveillance zone, lobby, is divided into three different zones: free zone, warning zone and restricted zone as illustrated in Fig. 4. It is preferable to place the camera within the restricted zone as illustrated in Fig. 1. As mentioned before, the depth plane size increases as it further away from the camera. Since the restricted zone in this system covers a smaller area, it is easier to define the boundary parallax if the camera is placed within this restricted zone.

The experiment is tested using an Intel Core2 CPU at 1.83GHz with 1.99GB of RAM. The input video is an AVI file with the resolution of 640x480.

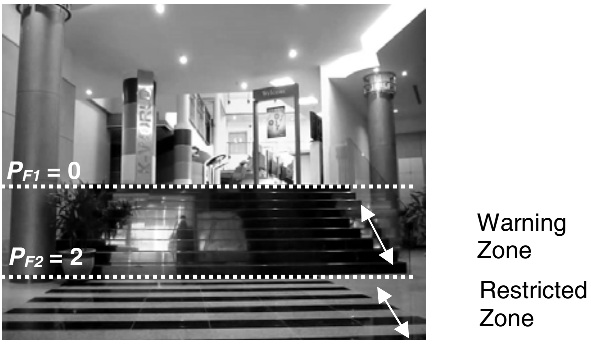


Fig. 4. Defining warning zone and restricted zone in a surveillance zone

4.2 Pre-processing

In this paper, the background model is the first image captured by the camera prior to the main algorithm starts. The scene is clear during the background model capturing.

As mentioned before, two virtual fences are defined in experiment presented in this paper and thus it requires four parallax values, P_n, as illustrated in Fig. 5. The boundary parallax values, P_F at the virtual fences are set to 0 and 2 whereas the parallax list values are {-1, 1, 3}. However, users are free to define the number of zone and the sensitivity of the program by modifying the P_F and P_n values.

5 Experimental Results

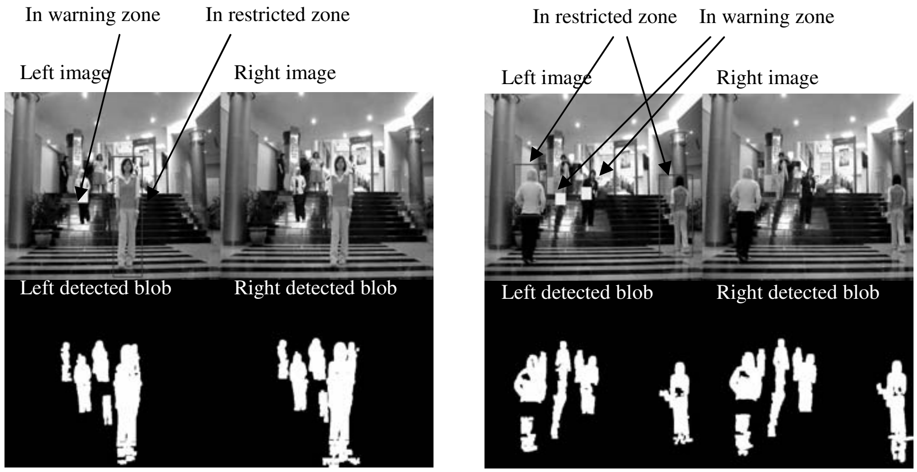
A few sets of experimental results are presented in this paper. The object that enters to the warning zone is indicated by a small solid box whereas the object that enters to the restricted zone is indicated by a bounding box.

More than thousands of video frames are experimented. Fig. 5 illustrates two sample video frames extracted from the experimental results.

Table 1 summaries some experimental results. The same set of image sequence is tested using three methods to determine the occupied zone for each person. The first method is the proposed algorithm with dense list of parallax values (P_n = {-3, -2, -1, 0, 1, 2, 3, 4, 5, 6}). The second method is the proposed algorithm with sparse list of

Table 1. Accuracy measurement and time taken for three experiments (total of 1000 frames)

Algorithm	Accuracy	Time taken	Frame rate
Dense parallax list	81.73%	91.45s	10.93fps
Sparse parallax list	84.52%	64.27s	15.56fps
Feature-based (Centroid)	60.99%	67.14s	14.89fps



(a) One object in restricted zone, one object in warning zone, two objects in free zone and an error blob due to reflection

(b) Two objects in restricted zone, two objects in warning zone, two objects in free zone.

Fig. 5. Extracted video frames from the experimental results with sparse parallax list

parallax values (P_n = {-1, 1, 3}) whereas the third method is using centroid to calculate its parallax value. The accuracy is calculated as follow:

$$\text{accuracy} = \frac{\#correct\ frame\ for\ \ person\,1+...+\#correct\ frame\ for\ \ person\,n}{\#\,frame\ \ for\ \ person\,1+...+\#\,frame\ \ for\ \ person\,n} x100 \quad (2)$$

6 Conclusion and Future Work

This paper presented a method to find the depth of objects of interest and more particularly to be applied in surveillance system. Using a simple background reference model, simple foreground extraction algorithm and image subtraction, the proposed method was able to perform well and provide acceptable results.

Although errors are detected, it can be further improved through future work by employing more robust foreground subtraction algorithm to provide more reliable input This can then be tightly integrated with the method for improved accuracy and detection.

References

1. Collins, R.T., Lipton, A.J., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., Hasegawa, O.: A System for Video Surveillance and Monitoring: VSAM Final Report, Technical report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University (2000)

2. Stefano, L.D., Marchionni, M., Mattoccia, S.: A Fast Area-based Stereo Matching Algorithm. In: Image and Vision Computing, vol. 22, pp. 983–1005 (2004)
3. Tombari, F., Mattoccia, S., Stefano, L.D.: Segmentation-based adaptive support for accurate stereo correspondence. In: Mery, D., Rueda, L. (eds.) PSIVT 2007. LNCS, vol. 4872, pp. 427–438. Springer, Heidelberg (2007)
4. Sun, C.M.: Fast Stereo Matching using rectangular Subregioning and 3D Maximum-surface Techniques. International Journal of computer Vision 47, 99–117 (2002)
5. Kaufman, P.L., Alm, A.: Adler's Physiology of the Eye, 10th edn. Elsevier Health Sciences, Amsterdam (2002)
6. Sekuler, R., Blake, R.: Perception, 3rd edn. McGraw-Hill, New York (1994)
7. Piccardi, M.: Background Subtraction Techniques: A Review. In: IEEE International Conference on System, Man and Cybernetics, pp. 3099–3104 (2004)

Application of mnSOM on Linking External Exposure to Internal Load

Stefan W. Roeder[1], Matthias Richter[1], and Olf Herbarth[2]

[1] Helmholtz Centre for Environmental Research - UFZ,
Department Human Exposure Research/Epidemiology
D - 04318 Leipzig, Germany
{stefan.roeder,matthias.richter}@ufz.de
[2] University of Leipzig, Faculty of Medicine,
D - 04103 Leipzig, Germany
olf.herbarth@medizin.uni-leipzig.de

Abstract. The internal load in humans caused by an external exposure is different in each person and mainly depends on metabolism. Using the recently proposed method of mnSOM we are able to describe the human metabolism using a functional module (linear or nonlinear) for each individual.mnSOM enables us to subdivide individuals into classes based on the functional description of each individuals metabolism. Furthermore the shown approach is able to show dependencies between external exposure and internal load in humans. In environmental epidemiology this will be used to establish links between external exposure and internal load patterns to gather clinical relevant information for practitioners.

1 Introduction

To gather deeper knowledge about human diseases potential influenced by exposures it is necessary to associate external and internal load patterns as well as binary outcome variables on the health state of individuals. The palette of external exposure patterns embraces environmental factors as well as socio-economic factors.

This paper describes an approach for associating external and internal load patterns in order to recognize patterns in these combined patterns. These patterns are used to deduce on the relationships and processes behind.

The main focus of this investigation is to classify individuals into several typical clusters of similar behaviour in terms of metabolism and subsequently to find core features of these clusters.

2 Material and Methods

The internal load resulting from an external (environmental) exposure depends on the metabolism capacity of the individual person. To gather deeper knowledge about these links, it is necessary to look behind the scenes and to find out which variables affect each other as well as which groups of individuals can be described.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 816–823, 2009.

Environmental exposure is described by variables like concentration of several chemical compounds, traffic intensity or socioeconomic parameters. They are gathered by measurement and through structured questionnaires.

Internal load patterns are described by both the resulting dose of the original substance and the metabolites which are generated during metabolism inside the human body. Information on human health is collected from physicians diagnose as well as from structured questionnaires.

Although internal load patterns contain a lot of useful information about the current state of the individual of origin, it is very difficult to handle these extremely large data sets with traditional statistical methods. The use of principal component analysis (PCA) is very common.

However, these methods assume a linear relationship between the measured sample and the converted principal component. Because in biological processes this linear relationship can not be assumed in general, errors are not controllable. Therefore it's crucial to have a method, which is able to map non-linear dynamics, in oder to represent the relationships and transformations in human metabolism. Furthermore, PCA requires prior definition of a particular number of factors, which is not known in most cases.

We decided to use a clustering method based on self organizing maps (SOM) [1;2]. They provide an efficient way to map from an n-dimensional space to a two-dimensional space and to visualize multivariate data. The self organizing map is a member of the class of the unsupervised artificial neural network algorithms. In contrast to supervised artificial neural network algorithms, self organizing maps are able to extract typical features without any prior knowledge about the structure of the data.

The extension mnSOM (modular network self-organizing map) proposed by Furukawa et al. [3;4] enhances the SOM with functional modules (FM) inside each neuron in order represent non-linear functions between independent and dependent variables (nonlinear input-output functions). as well as to map the objects under consideration (humans in our case) using the SOM algorithm. mnSOM enables us to subdivide individuals into classes based on the functional description of each individuals metabolism. Furthermore prediction of selected outcome variables as well as missing values is possible after training of the mnSOM network.

2.1 Application of mnSOM

The main advantage of self organizing maps is their ability to map a high-dimensional data set onto a lower dimensional (usually two-dimensional) space while preserving the original topological relationship between the objects in the data set. [1;5]. This advantage is used in mnSOM to map the objects onto a two dimensional space in order to show their similarities. Details of the original SOM algorithm can be found in [5] for theoretical considerations. The concept of mnSOM was first published by Furukawa et al. [6].

For mapping the dependencies of the metabolisms inside the human body, the functional modules multi layer perceptron (MLP) of the mnSOM are in use.

The mnSOM consists of two layers: the input layer and the Kohonen layer which holds the functional modules. Both layers are fully interconnected. The lattice type of the Kohonen layer can be taken as rectangular or hexagonal [5].

The topology of the MLP depends on the data structure under consideration. There is a input neuron needed for every variable in the environmental exposure vector. Also for the vector representing the internal load profile one neuron for every variable is needed.

Variables representing the exposure situation of an individual and genetic information, which can also be incorporated into the functional module, form the information on the input side of the functional module.

Information on human health as well as information on internal load forms the information on the output side of the functional module (dependent variables).

Sample data in the form

$$D_i = \{(x_{ij}, y_{ij})\}(i = 1,...,M(j = 1,...,N) \tag{1}$$

is known for each individual.

Using a MLP the internal relationships in each individual are trained into node weigths of the MLP's inside the functional modules. The metabolism as a transformation from environmental exposure to internal load can be described as a functional dependency of the internal load on the exposure scenario. Because of this functional dependency, we propose the use of MLP as functional module inside the mnSOM for mapping of the functional dependencies of human metabolism. These MLPs are trained using exposure variables as input and the internal load variables as output.

Using the mnSOM algorithm the similarities between the individuals can be represented inside the mnSOM after trainig. The result is a topology preserving mapping of all individuals onto a two-dimensional plane.

2.2 Normalization of Metric Variables

Metric variable values are normalized to a closed {0;1}-interval before processing [7]. This is done by the following assignment:

$$x_{ij}norm = \frac{x_{ij} - \min(x_j)}{\max(x_j) - \min(x_j)} \tag{2}$$

with i as the variable index and j as the case index. These combined data vectors are fed into a self organizing map. The number of functional modules in the mnSOM predicts the maximum number of clusters simultaneously.

After the training steps each functional module represents the centre of a cluster. Each cluster contains a typical set of cases with typical properties. These properties are best represented by the data vector of the functional module in the center of each cluster. Therefore the values of the data vectors of each functional module can be taken into account as best describing the environmental exposure situation in relationship to the outcome situation found in the cases of this cluster.

2.3 Training of the mnSOM and its Functional Modules

Exposure variables are fed into MLP of the functional modules as inputs; variables describing internal load as well as outcomes are fed into models as output vectors as shown in fig. 1. The link between is undefined at the beginning. After training of the

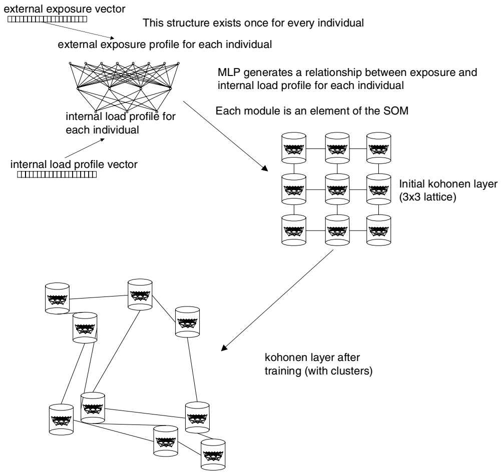

external exposure vector
This structure exists once for every individual
external exposure profile for each individual
MLP generates a relationship between exposure and internal load profile for each individual
Each module is an element of the SOM
internal load profile for each individual
internal load profile vector
Initial kohonen layer (3x3 lattice)
kohonen layer after training (with clusters)

Fig. 1. Overall system architecture

functional modules MLP the link is described by a functional dependency stored in the weights of each perceptron. According to Furukawa's idea, all functional modules are organized in the mnSOM.

The data set for training of the MLP is randomly chosen out of the dataset of all individuals at the beginning. All functional modules are trained by backpropagation.

The winner neuron in the mnSOM is selected by measuring the distance between the individuals data set and the set of functional modules inside the mnSOM. The functional module which is closest to the data set under consideration is trained by a higher learning rate, whereas functional modules farther away are trained by a lower learning rate. Therefore an adaptation of the mnSOM on the dataset is achieved. The mnSOM learning process is described in detail in [6]

The training process includes multiple training steps with the data set of this individual. Therefore the MLP fits very good to this single individual after training process. If an overfitting of the MLP happens, this doesn't matter, because overfitting comes along with a high accuracy of the fitted MLP to the individuals data set.

As mentioned above, mnSOM is capable to map from an n-dimensional attribute space to a lower dimensional (typically 2-dimensional) attribute space. To find class assignments it is necessary to add some calculations.

For clustering purposes we need a mnSOM with less nodes than individuals under consideration. So some of the nodes are able to map one or more individuals on them.

Class assignment for each individual can be determined by calculating the distance

$$L^2(\hat{O}(D_i), M^k) \tag{3}$$

of the dataset D_i against the weight vector of each module M^k after training using equation 4:

$$L^2(O_i, M^k) \cong \frac{1}{J}\sum_{j=1}^{J}(e_{i,j}^k)^2 \tag{4}$$

X and Y represent the weight vectors for measuring the distance in between. P stands for the number of dimensions of both vectors. $e_{i,j}^k$ denotes the error between data vector $r_{i,j}$ and the output of the k -th functional module [8]. Different metrics can be used [9].

The functional module with the least euclidean distance is the center of the class, the case in doubt belongs to. One can count the number of assigned cases for each Kohonen neuron. Kohonen neurons with large numbers of assigned cases represent class centers.

2.4 Estimation of Class Properties by Calculation of Parameter Values for Functional Modules

As proposed in [12] for the classical SOM approach it is also possible for the mnSOM to calculate the class properties of each class by calculation of parameter values for the functional module behind each class.

For the analysis of the results of clustering it is important to gather deeper knowledge about the variable values behind each class. as well as the relation of these values compared to all classes. Each class is represented by a functional module. Therefore all properties of a class are stored in the representing functional module.

As an extension to the classical SOM there is not only a weight vector, but there is also a functional dependency, described by e.g. a MLP in each functional module.

Because of the fact that the weight vectors of the functional modules contain normalized values in a closed {0;1}-interval these values must be 'denormalized' be before considering on them. Denormalization means the opposite transformation than normalization and transforms the values in the functional module back to the original co-domain of the underlying variable using the following equation:

$$x_{ij} = (x_{ij}norm * (\max(x_j) - \min(x_j)) + \min(x_j) \tag{5}$$

We use a visualization scheme for this purpose, which shows the values of a variable for all functional modules simultaneously. Each functional module is visualized by a bar, which is color coded and labeled with the denormalized value of the selected variable. Using these denormalized values the distribution of values can be inspected for each variable. The values behind functional modules which are not assigned as class centers are unimportant and can be removed from further inspection. Values behind functional modules representing class centers are at the same time the approximated values for the respective variable in this class.

2.5 Visualization of the Class Assignment

To estimate the goodness of classification, it is necessary to inspect the number of cases, which are assigned to each functional module. This approach is a modified version of the visualization of the U-matrix first described by Ultsch [10], who uses a

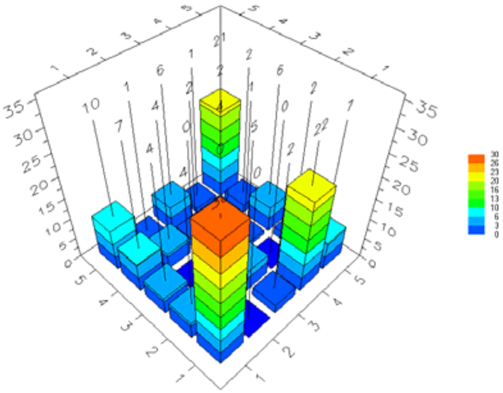


Fig. 2. Graphical representation of case assignment for a 5x5 Kohonen layer

two-dimensional visualization and colour coding of data density. The graphical representation of the case assignment is shown in figure 2.

The topological position of the functional module on the map is given by the numbers on x- and y-axis. The height of each bar is equal to the number of cases which are assigned to this neuron. Additionally the case-identifier and variable values behind each functional module can be obtained.

During several learning cycles a selected case can be assigned to different functional modules. This is a result of the random initialized weight vectors in the Kohonen layer and the subsequent evolving process. If the same group of cases is assigned to different functional modules this is no problem at all. It is also no big deal if several cases jump from one to another group. Difficulties arise if the group members change rapidly. Measuring of the permanence of class assignment of a single case over several learning cycles is necessary, but remains a task for future development.

Comparing clustering solutions generated by different mnSOM topologies with possibly different class numbers can be done by applying Davies Bouldin Index [11].

2.6 Prediction of Variable Values

Variable value prediction can be subdivided into prediction of missing variable values and prediction of future outcome variables.

Most data sets in epidemiology do have missing values. Using a trained mnSOM, it is possible to calculate the most probable value for the missing variable value out of the weights of the functional module which is closest to the case under consideration.

Another area for application the proposed mnSOM architecture is the prediction of future outcome variables. This is very close to classification, because it will enable us to predict future health state of study participants from the already present measurements. It is necessary, that outcome variables for at least some cases are known.

For prediction of outcome variables we trained the self organizing map with cases which outcomes were known. In a second step the unknown cases for prediction were fed into the map. The Euclidean distance against each Kohonen neuron is calculated as described above. The Kohonen neuron with the least distance represents the best matching scenario. Its value for the outcome variable under consideration is the most probable future value for the selected case.

For evaluation purposes the calculation of a correct prediction rate is essential. This can be done by dividing a given data set into a training set and a test set which are nearly equal sized. The training set is used to train the SOM to a stable state. Afterwards the test set is used to verify the outcomes prognosed by the SOM. During the forecasting process it is important not to use the outcome variables for calculation of the euclidean distance. This means with n variables (dimensions) and k outcome variables to predict, the euclidean distance is only calculated from the n-k remaining variables.

3 Results

The approach presented here, integrates data vector normalization, MLP training, mnSOM training, class finding, visualization and variable prediction.

In a first step we were able to determine settings for mnSOM parameters, which guided us to a stable configuration. We were able to clearly see the fact, that the mnSOM algorithm does not force all functional modules to be populated with data from individuals. Consequently there is an upper limit for the number of functional modules in each dimension of the mnSOM. We examined this limit by visual inspection and gradually lowering the number of functional modules in each dimension of the mnSOM.

If the number of functional modules is too high, then a significant number of modules remains unassigned. These unassigned modules can be perceived as borders between the classes. Simultaneously the number of assigned cases to each assigned functional module is relatively low. Typically one can observe several contiguous functional modules with only slight differences. By reducing the number of functional modules these minor differing neurons will be summarized into one module representing the respective class. This process of reconfiguration and visual inspection is supported by indicators for classification quality such as Davies-Bouldin-Index. Future planning includes automation of this process using computer grid technology.

4 Discussion

Using this approach we are able to

a) recognize dependencies between external exposure and internal load situation using mnSOM with the functional module MLP,

b) cluster individuals into groups and recognize similarities between them,

c) predict missing values,

d) predict the probabilistic value of an outcome variable for a given environmental load situation.

In environmental epidemiology this will be used to establish links between external exposure and internal load patterns to gather clinical relevant information for practitioners.

In consequence the shown approach is easy to handle and does not require complex equipment for analysis. It can therefore be established in nearly every laboratory provided with state-of-the-art computational equipment.

Examinations with data from different studies realized at our department have shown the usefulness of this approach. Future plannings include the application of this method to data from longitudinal studies from birth cohorts to gain knowledge about critical time windows in childhood development.

References

1. Kohonen, T.: Self Organizing Maps. Series in Computer Sciences edn. Springer, Heidelberg (1997)
2. Zampighi, L.M., Kavanau, C.L., Zampighi, C.A.: The Kohonen self-organizing map: a tool for the clustering and alignment of single particles imaged using random conical tilt. Journal of Structural Biology 146, 368–380 (2005)
3. Tokunaga, K., Furukawa, T.: Modular Network SOM: Theory, Algorithm and Applications. In: King, I., Wang, J., Chan, L.-W., Wang, D. (eds.) ICONIP 2006. LNCS, vol. 4232, pp. 958–967. Springer, Heidelberg (2006)
4. Tokanuga, K., Furukawa, T., Yasui, S.: Modular network SOM: Extension of SOM to the realm of function space. In: Proceedings of WSOM 2003, pp. 173–178 (2003)
5. Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer, New York (2000)
6. Tokanuga, K., Furukawa, T.: Generalized Self-Organizing Maps (mnSOM) for Dealing with Dynamical Systems. In: Proceedings of the 2004 International Symposium on Nonlinear Theory and its Applications (NOLTA 2004), pp. 231–234 (2004)
7. Zupan, J., Gasteiger, J.: Neural Networks in Chemistry and Drug Design. Wiley-VCH, Weinheim (1999)
8. Tokanuga, K., Furukawa, T.: Generalization of the Self-Organizing Map: From Artificial Neural Networks to Artificial Cortexes. In: King, I., Wang, J., Chan, L.-W., Wang, D. (eds.) ICONIP 2006. LNCS, vol. 4232, pp. 943–949. Springer, Heidelberg (2006)
9. Zell, A.: Simulation neuronaler Netze. Oldenbourg, München (2000)
10. Ultsch, A.: Self-organizing Neural Networks for Visualization and Classification. In: Opitz, O., Lausen, B., Klar, R. (eds.) Information and Classification, pp. 307–313. Springer, Berlin (1993)
11. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. Pattern Anal. Machine Intell. 1(4), 224–227 (1979)
12. Roeder, S.W., Rolle-Kampczyk, U., Herbarth, O.: Visualization of Depending Patterns in Metabonomics. In: King, I., Wang, J., Chan, L.-W., Wang, D. (eds.) ICONIP 2006. LNCS, vol. 4234, pp. 278–284. Springer, Heidelberg (2006)

Part VI

Neuromorphic Hardware and Embedded Neural Networks

Automated and Holistic Design of Intelligent and Distributed Integrated Sensor Systems with Self-x Properties for Applications in Vision, Robotics, Smart Environments, and Culinary Assistance Systems

Andreas König

Institute of Integrated Sensor Systems, TU Kaiserslautern
67663 Kaiserslautern, Germany
koenig@eit.uni-kl.de

Abstract. The ongoing advance in micro technologies gives rise to increasingly versatile and capable sensors as well as unprecedented computational power and communication options in diminishing scale. The notion of smart dust summarizes ubiquitous computing and sensing application systems, which can serve for local as well as global information acquisition and decision making. For off-the-shelf-nodes, and even more for dedicated physical designs, the system design process becomes increasingly challenging and potentially intractable. Automated design methods emerging for intelligent systems are introduced as a remedy. These considerations will be extended to variations, that multiple system instances have to face in real-world applications and potential compensation by incorporation of self-x properties. These concepts are elucidated for the case of reconfigurable and evolvable sensor electronics. Finally, an application perspective of the presented approach for integrated distributed sensing in home automation, assisted living, and in particular, smart kitchen applications, denoted as culinary assistance systems will be presented.

1 Introduction

The incessant advance in micro and nano technologies paves the way to increasingly versatile and capable sensors, powerful communication capability, abundant computational power, efficient energy harvesting schemes, and affordable MEMS system integration. Complementing these physical benefits with the power of computational intelligence (IMEMS) gives rise to innovative solutions for many applications, e.g., in vision, robotics [3], biometrics, automotive, and automation [1,2]. The design of such systems, even restricted to off-the-shelf (OTS) components, imposes a severe load on the designers. Scene, sensors, and methods choice and parameterization are a tedious and time-consuming task, requiring expert skills and commonly producing moderate quality results at high cost and design effort (**Fig.1a**)).

2 Intelligent System Design Automation

Design automation techniques and a holistic view are salient here (**Fig. 1** b)) and a methodology with corresponding tool implementation is pursued in our research for

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 827–834, 2009.

vision tasks, , e.g., visual inspection, texture analysis, and other sensor applications for more than a decade. Our QuickCog system (**Fig. 1** c)) [6] was our first framework to implement the devised architecture. Automation was limited to the level of dimensionality reduction and classification. Other design activities were interactively supported, e.g., based on effective data visualization. The architecture has been extended to feature computation level for image processing and texture analysis [7] and to multi-sensor system design, e.g., for gas sensor data analysis or automotive tasks [8,9]. The research comprised development and application of suitable assess-ment measures and optimization strategies, e.g., support-vector-machines (SVM) and particle swarm optimization (PSO). The results in image and sensor applications showed, that our design automation approach helps less skilled users to find practical solutions with less effort and even can beat expert solutions. In more recent activities multi-objective optimization is investigated to achieve well performing but lean systems, i.e., resource-aware realizations for embedded/integrated sensor systems.

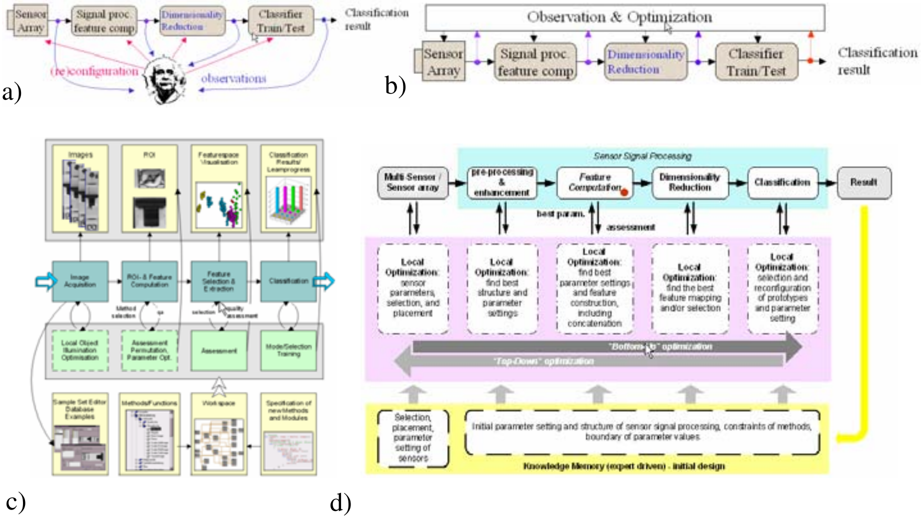

Sensor Array
Signal proc. feature comp
Dimensionality Reduction
Classifier Train/Test
Classification result
(re)configuration
observations
a)
Observation & Optimization
Sensor Array
Signal proc. feature comp
Dimensionality Reduction
Classifier Train/Test
Classification result
b)
Images
ROI
Featurespace Visualisation
Classification Results/ Learnprogress
Image Acquisition
ROI- & Feature Computation
Feature Selection & Extraction
Classification
Method selection
selection
quality assessment
Local Object Illumination Optimisation
Assessment Permutation, Parameter Opt.
Assessment
Mode/Selection Training
Sample Set Editor Database Examples
Methods/Functions
Workspace
Specification of new Methods and Modules
c)
Sensor Signal Processing
Multi Sensor / Sensor array
pre-processing & enhancement
Feature Computation
Dimensionality Reduction
Classification
Result
best param.
assessment
Local Optimization: sensor parameters, selection, and placement
Local Optimization: find best structure and parameter settings
Local Optimization: find best parameter settings and feature construction, including concatenation
Local Optimization: find the best feature mapping and/or selection
Local Optimization: selection and reconfiguration of prototypes and parameter setting
"Bottom-Up" optimization
"Top-Down" optimization
Selection, placement, parameter setting of sensors
Initial parameter setting and structure of sensor signal processing, constraints of methods, boundary of parameter values
Knowledge Memory (expert driven) - initial design
d)

Fig. 1. Issues of and architectures for intelligent system design automation

3 Instance-Specific Adaptation in Multi-Site Deployment

While visual inspection systems commonly feature small lot sizes, e.g., lot size one, in robotics or driver or live assistance systems large lot sizes can be expected, giving rise to special considerations on instance specific deviation compensation and utmost resource-efficiency (**Fig. 2a)**), motivating methodology extensions. Inspired by concepts of intrinsic evolution in microelectronics and evolvable hardware, Machine-In-the-Loop-Learning (**Fig. 2b)**) [3] has been introduced to achieve adaptation for compensation and robustness increase as well as to some degree fault-tolerance and applied to the case of a commercial laboratory robot system (**Fig. 2c)**). In this work, the initially devised system, based on the techniques introduced in the previous

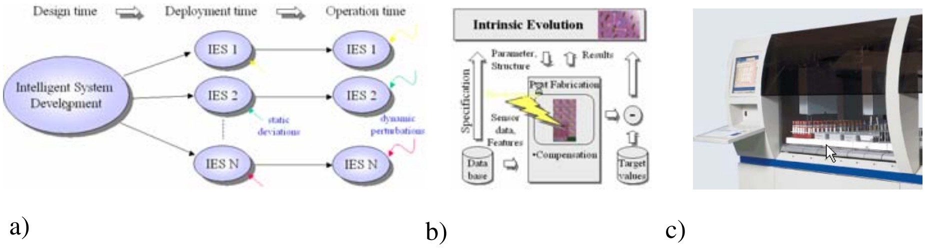


Fig. 2. Intelligent system performance preservation for multi-site deployment in medical robot

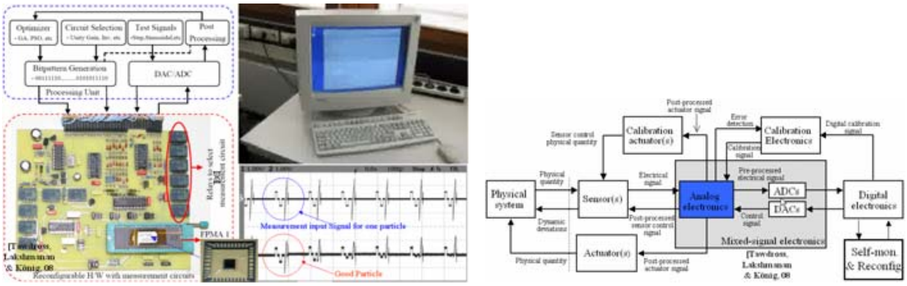
section, is kept re-trainable so that site-specific deviations or aging effects can be compensated [3]. The approach is well accepted in industrial use and sees extension.

4 Inclusion of Self-x Properties

Further needed increase of system flexibility, robustness, and fault-tolerance will be added by adaptive, self-x features on the hardware level itself. Our research deals with reconfigurable and evolvable sensor electronics (**Fig. 3**)). A redundant analog array was designed in this work, that can be reconfigured by programmable switches. The switch patterns can be adapted or optimized under the control of, e.g., a PSO algorithm. For the special case of sensor signal conditioning special chips and system prototype have been developed [10-12]. In intrinsic, multi-objective evolution the hardware performance can be tuned and retained in the face of static or dynamic influences and deviations. **Fig.3 a)** shows achieved intrinsically evolving sensor electronics and **Fig.3 b)** the aspired system concept, that complements the approach of section 3 for improved robustness and fault-tolerance in intelligent systems.

5 Extension to SmE, AmI, and Culinary Assistance Systems

The ongoing advance in Micro/Nano-technologies, commonly elucidated by Moore's law, adds unprecedented computational power, capable sensors in rich variations, and (wireless) communication options. Distributed sensing based on cooperating (multi) sensor nodes offers the baseline for more powerful smart environments. Smart or sensate floors, e.g., based on capacitive sensor principles [4], are one prominent example of distributed, large area sensing, contrasting the diminishing sensor sizes in MEMS, and provide context awareness in many applications. However, sensor technology is heterogeneous & differs from main stream (CMOS) integration activities. So, the common extrapolation of Moore's law to this domain is not fully justified. Collectively collaborating sensing and processing nodes are the baseline for SmE, such as smart homes (home automation), smart offices, smart hospitals/health care, smart cars (Automotive, Driver-Assistance-Systems, engine monitoring and control, ..), smart streets (traffic monitoring and control), smart factories, smart classrooms (E-Choke), or smart shops. Such new application domains add complexity, challenges, and opportunities. The extension of our described design approach to such distributed, networked systems of ubiquitous computing, SmE, and AmI based on IMEMSis



a)

b)

Fig. 3. a) Intrinsically evolvable sensor electronics, b) evolvable concept for complete system

currently pursued in our research to answer the emerging design challenges for collective and holistic intelligent system design. This is particularly demanding, as the cross-disciplinary integration of skills and technologies from artificial/computational intelligence, human-computer-interaction, sensor & actuators, power consumption minimization and energy harvesting, chip/MEMS-based system integration as well as advanced communication and networking techniques is required. The field of AmI (or ubiquitous computing) has received considerable attention in the last five years and is pursued in numerous national and international programs. A common AmI-definition is: "*An intelligent & adaptive electronic environment that proactively, but sensibly assists people in their daily life*" [1]. This includes the aspects of enhancing human abilities (General Assistance-Systems), restoring & preserving human abilities (assisted-living/home care), unobtrusive systems (often today's off-the-shelf solutions are insufficient for this aim, requiring dedicated solutions, .e.g., MEMS), networked systems (wireless, wired (KNX-bus, or power grid)), context-aware systems (requiring complex object/ event recognition & tracking capability), personalizable, adaptive & anticipatory systems (dito !), as well as robust, reliable & low-power autonomous operation. In some activities in excess to body-areas-networks, commonly related to RF-ID or medical activities, implants in the human body are advocated by the community. The vision of such an emerging cyborg is reminiscent of a well-known science-fiction plot of StarTrek. The borg civilization introduced there gives a warning of potential loss of privacy and even individuality and control due to the omnipresent scrutiny by such systems. Clearly, new rules and practices have to be established to avoid abuse and infringement of human rights by the entering in home and body of this more and more powerful technology.Research on Smart Homes or Intelligent Houses unifies automation and AmI R&D, with the focus on increasing comfort, energy efficiency, safety and last not least to provide life assistance to challenged or elderly people (AAL). World-wide numerous activities can be found, e.g., the Fraunhofer IMS Duisburg intelligent house, the GatorTech smart house, Samsung Hauzen, Philipps home lab, NTT com lab world of mushrooms, AmI lab at the University of Madrid, or Microsoft lab can be named as prominent representatives. As in industrial automation scenarios, the human living environment, house or apartment, is instrumented, i.e., equipped with sensors and processing/communication facilities commonly backed up by a central automation server and software. Again, it seems fancy inspirations from the world of movies have come to life, if numerous activities are compared, e.g., with the devices and automation state of the home of famous *Wallace and Grommit* characters. Due to its cultural and social importance and technical challenges, the kitchen has received special attention in the last years. An early activity of DIAS GmbH, Dresden, and Siemens, designing an IR-sensor-based overcook monitoring system can be mentioned here [16]. Prominent recent activities are the Counter Intelligence project by MIT [2], TU Munich's smart kitchen [5], General Electrics kitchen of the future, IBM's smart kitchen, or DFKI's Shared Life project activities [14,15]. Smartness of existing devices, e.g., refrigerator or dishwasher, is enhanced by additive intelligent sensing systems or new devices and functionalities are conceived, e.g., MIT's heat sink, up-down sink, smart spoon, and graphical user interfaces [2], TU Munich's smart knife [5], DFKI's intelligent fridge and semantic cookbook, or LMU's living cookbook. In many cases, known approaches from professional food production and processing in industry and restaurants are just adapted for home use. For instance the established commercial ChefTec Software of CSS offers numerous features of extendable hierarchical recipe database and annotation options by recording of cooking activities along with storage and utensils management and nutrintional analysis for quite some time now [13]. Summarizing,, current activities predominantly rely on networked off-the-shelf sensor & RFID tags, hardware & automation equipment. A strong emphasis on multi-media & networking with conventional home automation in newly-raised buildings can be observed. Definitely, the low hanging fruits in smart kitchen activities have been collected and promising ideas, e.g., MIT's heat sink, have already found commercialization. However, numerous issues still can be raised, that demand for more sophisticated research activities. The issue of actual intelligent, appreciated proactive and also feasible behavior tentatively is illustrated by **Fig. 4 a)** for the notion of an *intelligent* fridge.

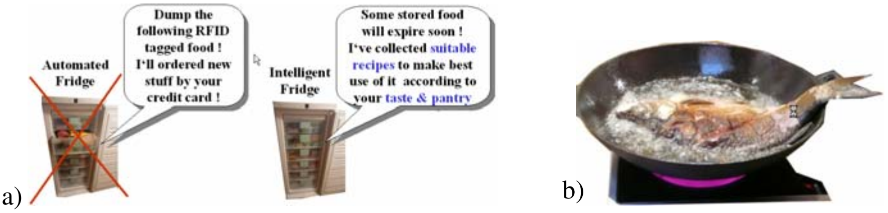

Automated Fridge
Dump the following RFID tagged food ! I'll ordered new stuff by your credit card !
Intelligent Fridge
Some stored food will expire soon ! I've collected suitable recipes to make best use of it according to your taste & pantry
a)
b)

Fig. 4. a) Issues of intelligent, appreciated behavior and b) perception in smart kitchens

A further issue is, that the majority of homes is *seasoned,* which requires special effort for cost effective integration of concepts and systems in the existing environment. Wireless (NFC/RFID) or power-line based communication offer feasible options. A key issue, however, is the fact, that successful concepts based on RFID tagging cannot be so easily copied from industry to home. Fresh ingredients and home made food (sub recipes) are not RFID tagged, i.e., require more capable integrated sensing and recognition.

This gives rise to a research activity and the notion of Culinary Assistance Systems (CAS), targeting on average home inhabitants' support with regard to comfort, lifestyle improvement, economy, and security, by augmenting sensory, cognitive, and organizational skills. CAS are hardware/software hybrids which shall provide economic use of resources (food, energy, etc.), extend quality of life (chef experience to the home), and augment and preserve sensing and assessment capabilities. The last feature requires CAS to be able to deal with arbitrary appearance independent of RFID tags, to detect potential food pollution and decay, to detect potentially dangerous ingredients/contaminants, and to assist in better understanding and measuring of dynamic preparation states, depending on volume V, (distributed) temperature T, color, and other information, by multi-sensor annotation. Clearly, CAS shall serve to deal with hard multi-sensorial recognition as well as embedding/integration problems (IMEMS).

These intelligent system design challenges for various tasks of food storage, control, and preparation stimulate our methodology's extension for distributed sensing by networked OTS system or self-x IMEMS.

One simple example of a CAS will be presented in the following. Mushrooms are a delicacy esteemed in cuisines world-wide. However, classification of self-collected mushrooms from the woods and meadows into poisonous and edible ones as well as assessment of the state with regard to freshness or decay is a challenging task requiring either expert knowledge or appropriate assistance (see **Fig. 5**).

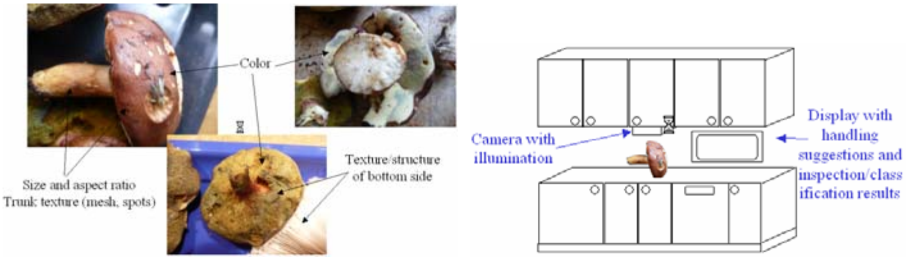
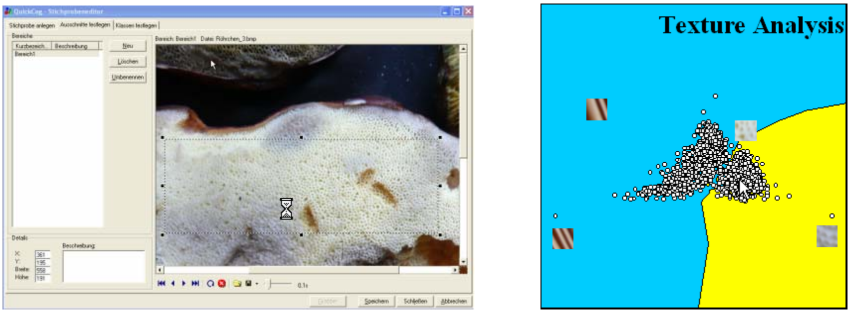

Color
Size and aspect ratio
Trunk: texture (mesh, spots)
Texture/structure of bottom side
Camera with illumination
Display with handling suggestions and inspection/class ification results

Fig. 5. Concept of mushroom identification CAS

Numerous static classification features, some time-dependent ones, e.g., cut trunk and see, whether it turns bluish, black, or is unchanged. Thus, interaction is required in the analysis process. The CAS will alleviate interactive inspection of features according to database of expert knowledge. The knowledge of literally thousands of mushroom/toadstool species and instance variations, commonly reflected in books, will be incorporated in the CAS and made available to the home. MushID CAS is an applicable and needful example for intelligent system design automation, e.g., the texture analysis approach of section 2 can serve both for type and state identification (see **Fig. 6**). This straightforward CAS can be achieved with standard camera, but other similar tasks, e.g., measuring temperature profiles or detecting surface/skin color for perfect crispy but unscorched meat, poultry, or fish in roasting or frying processes or other examples. Many more could be conceived in food preparation processes. In particular, state-of-the-art video recording of cooking processes will be some help in simple cases, but will be insufficient for proper repetition, because important information will be missing, e.g., due to video sensor limitation (s. **Fig. 4b)**) and changing context. Additional sensorial context and sophisticated recognition/ assessment will be needed to support unskilled as well as challenged persons for a safe, more affordable, and better life.



Bereich: Bereich1 Datei: Röhrchen_3.bmp

Bereich1

X: 361
Y: 195
Breite: 558
Höhe: 191

Fig. 6. Mushroom classification based on texture analysis

6 Conclusions

The paper motivated the technological as well as application potential of intelligent systems, in particular focusing on underlying design challenges which can be met by the pursued design automation. The regarded approach and its extension to achieve improved robustness by the introduction of self-x capabilities in soft- and hardware was summarized for vision and other sensorial modes. A new rich application field in the context of SmE/AmI was presented and a first example for the proposed Culinary-Assistance-Systems was given. In our future work, merging long term engineering and culinary experience and interests, the CAS concept will be pursued in smart kitchen context to restore, preserve, and enhance human abilities both to bring chef quality to the home, safe resources, give safety, and keep cost in check.

Acknowledgments. The contributions of K. Iswandy, S. K. Lakshmanan, P. M. Tawdross, M. Eberhardt, C. Fei, S. Peters, and A. Chandra, to this work are gratefully acknowledged.

References

1. Augusto, J.C.: Ambient Assisted Living (AAL): Past, Present and Future. In: 2nd Workshop on Behavior Monitoring and Interpretation BMI 2008 at KI 2008, Kaiserslautern (2008)
2. Bonanni, L., Lee, C.H., Selker, T.: Counter Intelligence: Augmented Reality Kitchen. In: CHI 2005, Portland, Oregon, USA (2005)
3. Eberhardt, M., Roth, S., König, A.: Industrial Application of Machine-In-the-Loop-Learning for a Medical Robot Vision System - Concept and Comprehensive Field Study. In: Advances on Computer-based Biological Signal Processing (CEE), Elsevier, Amsterdam (2008)
4. Steinhage, A., Lauterbach, C.: Monitoring Movement Behavior by Means of a large Area Proximity Sensor Array in the Floor. In: 2nd Workshop on Behavior Monitoring and Interpretation BMI 2008 at KI 2008, Kaiserslautern (2008)
5. Kranz, M., Schmidt, S., Rusu, R.B., Maldonado, A., Beetz, M., Hörnler, B., Rigoll, G.: Sensing Technologies and the Player Middleware for Context Awareness in Kitchen Environments. In: 4th Int. Conf. on Networked Sensing Systems, June 6-8, 2007, pp. 179–186 (2007)

6. König, A., Eberhardt, M., Wenzel, R.: QuickCog Self-Learning Recognition System - Exploiting machine learning techniques for transparent and fast industrial recognition system design. In: Image Processing Europe, PennWell, September/October Issue, pp. 10–19 (1999)
7. Peters, S., König, A.: Optimized texture operators for the automated design of image analysis systems: Non-linear and oriented kernels vs. gray value co-occurrence matrices. Special issue of International Journal of Hybrid Intelligent Systems, IJHIS (2007)
8. Iswandy, K., König, A., Fricke, T., Baumbach, M., Schütze, A.: Towards Automated Configuration of Multi-Sensor Systems Using Evolutionary Computation - A Method and a Case Study. Journal of Computational and Theoretical Nanoscience 2(4), 574–582 (2005)
9. Iswandy, K., König, A.: A Novel Fully Evolved Kernel Method for Feature Computation from Multisensor Signal Using Evolutionary Algorithms. In: Proc. of 7th International Conference on Hybrid Intelligent Systems (HIS 2007), Kaiserslautern, Germany (2007)
10. Tawdross, P., König, A.: Mixtrinsic Multi-Objective Reconfiguration of Evolvable Sensor Electronics. In: Proceedings of Second NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007), Edinburgh, Scotland, UK, pp. 51–57. IEEE Computer Society, Los Alamitos (2007)
11. König, A., Lakshmanan, S.K., Tawdross, P.: Concept and First Evaluation of Dynamically Reconfigurable Sensor Electronics. In: Proc. of the 13th Int. Conf. Sensor Conference 2007, SENSOR+TEST 2007, Nürnberg, Germany, May 22-24, 2007, pp. 277–282 (2007)
12. Lakshmanan, S.K., Tawdross, P., König, A.: Towards Generic On-the-Fly Reconfigurable Sensor Electronics for Embedded System - First Measurement Results of Reconfigurable Folded Cascode Amplifier Building Block. In: Proc. of the 20th Int. Conf. on VLSI Design, Bangalore, India, January 6-10 (2007)
13. ChefTec, Product Information (2008), `http://www.culinarysoftware.com`
14. Schneider, M.: The Semantic Cookbook – Sharing Cooking Experience in the Smart Kitchen. In: 3rd IET Int. Conf. on Intelligent Environments, pp. 416–423 (2007), `http://www.dfki.de/web/shared-life`
15. Wahlster, W.: Das Internet der Dinge im Alltag – Umgebungsintelligenz rund um das Auto und den Supermarkt der Zukunft. Presentation at Tag der Technologie 2007, November 22, Fraunhoferzentrum, Kaiserslautern (2007)

Hardware Design of Japanese Hand Sign Recognition System

Hiroomi Hikawa[1] and Hirotada Fujimura[2]

[1] Kansai University, Suita-shi Osaka 564-8680 Japan
[2] Genesis Technology Inc., Nishiwaki-shi, Hyougo, 677-0052 Japan

Abstract. This paper discusses the hardware design and implementation of a hand sign recognition system with a simplified discrete Fourier transforms (DFTs) that calculate the magnitude spectrum. Two alternative hardware design solutions that implement the system are proposed. One uses parallel classifier network, the other uses serial one. With the parallel network, the circuit size of the recognition system is over 280,000-gate while the system with the serial classifier network requires about 90,000-gate of hardware resources. Regarding the operating speed, it has been revealed that the operation speed of the both system is quick enough to process NTSC video frame in real time.

1 Introduction

The use of hand gesture provides an attractive alternative to cumbersome interface devices for human-computer interaction (HCI) and many hand gesture recognition have grown in recent years [1]. Generally hand gestures are either static hand postures[2][3] or dynamic hand gestures[4][5].

In [6], hardware friendly hand sign recognition system has been proposed. In the system, input images are preprocessed through horizontal/vertical histogram calculations followed by discrete Fourier transform (DFT) that calculate the magnitude spectrum, which is used as the feature vector. Use of the magnitude spectrum makes the system very robust against the position changes of the hand image. Our objective is to develop a hardware-based posture classification system, but the use of DFT is not suitable for the hardware implementation. In order to reduce the circuit size, another hand sign recognition system with a simplified DFT is proposed in [7].

This paper is focused on the hardware design of the recognition system and two alternative solutions that implement the classifier network have been proposed.

2 Hand Posture Recognition System

Input image is $P \times Q$ pixels, RGB color format and the input image is preprocessed to obtain feature vectors. The feature vector is fed to the classifier network which finally identifies the hand sign.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 835–842, 2009.

2.1 Preprocessing

First, the input color image is converted to a binary image. The proposed system requires users to wear a red glove so that the background image is removed and the hand portion can be extracted easily. The extraction and binary quantization is done by,

$$I(x,y) = g(\ Red(x,y), Green(x,y) + Blue(x,y)\ ) \cdot g(\ Red(x,y), \rho\ ) \tag{1}$$

where, $I(x,y)$ is the binary pixel value, while $Red(x,y)$, $Green(x,y)$ and $Blue(x,y)$ are red, green and blue color levels at (x,y), respectively. ρ is a threshold parameter and $g(\cdot)$ is a threshold function. $I(x,y)$ is then used to obtain horizontal and vertical histograms, $P_H(y)$ and $P_V(x)$.

$$P_H(y) = \sum_{x=0}^{P-1} I(x,y), \quad P_V(x) = \sum_{y=0}^{Q-1} I(x,y) \tag{2}$$

Ten the DFTs convert $P_H(y)$ and $P_V(x)$ into the magnitude spectrums $F_H(n)$ and $F_V(n)$, respectively. $F_H(n)$ and $F_V(n)$ of the same hand posture images placed in different positions are identical, therefore the system is very robust against the position change between the training and input images. However, the conventional DFT is not suitable for hardware implementation as they include complex functions and multiply operations. To simplify the equations, cos(·) and sin(·) functions are replaced by tcos(·) and tsin(·), respectively.

$$\hat{A}(k) = \sum_{n=0}^{N-1} x(n) \cdot \mathrm{tcos}(\frac{2\pi nk}{N}) \tag{3}$$

$$\hat{B}(k) = \sum_{n=0}^{N-1} x(n) \cdot \sin(\frac{2\pi nk}{N}) \tag{4}$$

As shown in Fig. 1, tcos(·) and tsin(·) are the heavily quantized (tri-state) version of cos(·) sin(·) functions taking only three values, i.e., −1, 0 or +1. The calculation of the magnitude is also simplified as,

$$\hat{X}(k)\ =\ |\ \hat{A}(k)\ |\ +\ |\ \hat{B}(k)\ | \tag{5}$$

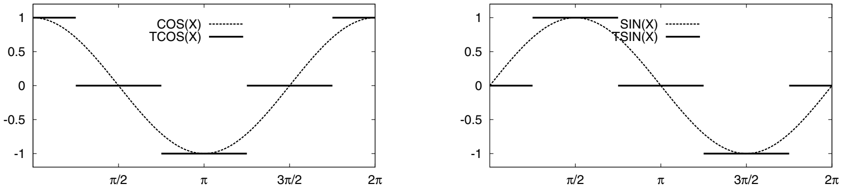


Fig. 1. Tri-state tcos, tsin functions

2.2 Classifier Network

The input to the network is a D-dimensional vector $\boldsymbol{x}$ from the preprocessing, which is fed to all neurons. Each input vector element x_i is taken from $F_H(n)$ and $F_V(n)$,

$$x_i = \begin{cases} F_H(i) & 0 \leq i < D/2 \\ F_V(i - D/2) & D/2 \leq i < D \end{cases} \tag{6}$$

Each neuron is associated with one of the hand sign classes, and it evaluates if the input vector is in their assigned sign. The neuron output $E^{(s)}$ is,

$$E^{(s)} = \sum_{m=1}^{M} \sum_{n=1}^{D} r_{nm}^{(s)}(\ x_n\ ) \cdot w_m \tag{7}$$

where, w_{nm} is a weight, and $r_{nm}^{(s)}(x_n)$ is range test function,

$$r_{nm}^{(s)}(x_n) = \begin{cases} 1 \text{ if } \quad U_{nm}^{(s)} > x_n > L_{nm}^{(s)} \\ 0 \text{ otherwise} \end{cases} \tag{8}$$

$U_{nm}^{(s)}$ and $L_{nm}^{(s)}$ are upper lower limits of the range in which the vector element $x_n^{(s)}$ is expected to be in, where s denotes the hand sign class. M upper and lower limit sets are defined for each vector element. $U_{nm}^{(s)} = \mu_n^{(s)} + \alpha_m \cdot \sigma_n^{(s)}$, $L_{nm}^{(s)} = \mu_n^{(s)} - \alpha_m \cdot \sigma_n^{(s)}$, $m = 1, 2, \cdots, M$. $\mu_n^{(s)}$ and $\sigma_n^{(s)}$ are the mean and standard deviation of the n-th vector element in the training vectors belonging to class s. α_m is a coefficients to adjust the upper and lower limits. The weight value in eq. (7) is determined by considering the size of the range.

Each neuron performs eq. (7). Winner-takes-all competition by the maximum value finder circuit is employed for the final classification.

3 Simulation

Computer simulations were conducted to verify the feasibility of the system. With the preliminary test, we decided to use the following parameters, $\rho = 150$, $M = 2$, $\alpha_1 = 1.0$, $\alpha_2 = 0.4$, $w_1 = 1$ $w_2 = 3$ and $D = 22$.

3.1 Effect of the Simplified DFT on the Recognition

The effect of the new DFT was investigated by using hand images made of 41 classes, each of which consists of 100 images. The data set is made of two groups, LT and RB groups. The LT group consists of images in left-top corner of the frame, while the RB have only images in right-bottom corner. The difference among images belonging to the same class is their positions, and their hand shapes are identical. For training the network, three types of learning data, RB, LT, MIX are used.

RB : randomly selected 20 images from RB group,

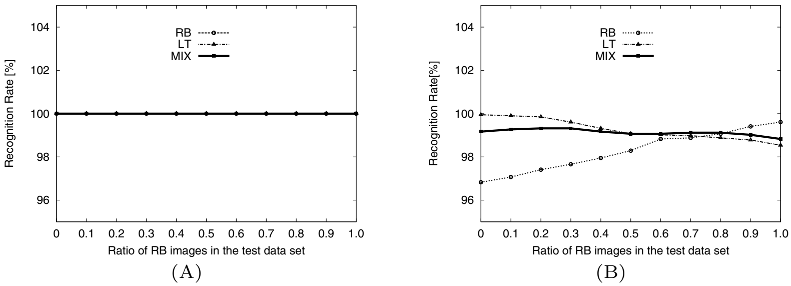


Fig. 2. Position change vs. recognition rate, (A) with conventional DFT, (B) with new DFT

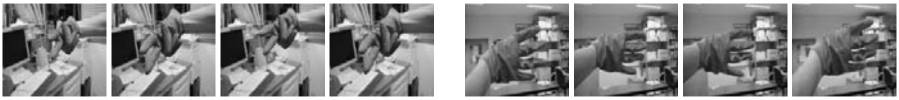


Fig. 3. Input image examples

LT : randomly selected 20 images from LT group,
MIX : 10 images are randomly selected from the both groups.

Recognition rates were obtained using test data sets with different ratio of RB group images. Fig. 2(A) shows that the recognition rate of the system with conventional DFT is 100%. This is because the effect of the position difference is removed by the DFT. Fig. 2(B) shows the relation between the recognition rate of the new DFT and P, which shows the correlation with the input data and the training data set. Trained with RB or LT data set, the worst recognition rate is about 97%, while the recognition rate of the system trained with the MIX data set, deterioration is only 1%.

3.2 Recognition of 41 Hand Signs

The proposed system was tested by 41 static Japanese hand signs. Some examples used for the experiment are shown in Fig. 3. As this figure shows, the sizes and shapes of the images are different even though they belong to the same class.

The results is summarized in Table 1. As the images used in the training are different in positions and shapes, the difference between the recognition performance between the systems with conventional and new DFTs is only about 1% as the previous simulation indicated.

Table 1. Average recognition rates

	system with conventional DFT	system with new DFT
Average recognition rate(%)	93.46	92.49

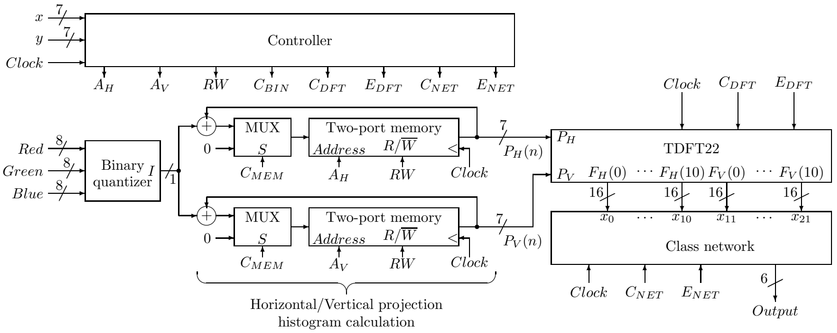


Fig. 4. Block diagram of the hand sign recognition system

4 Hardware Design

4.1 System Configuration

Fig. 4 shows the block diagram of the hand posture recognition system consisting of a binary quantizer, two memories, TDFT22 that performs simplified 22-point DFT, the classifier network and signal generator. The input signal are the RGB color signals ($Red, Green, Blue$) of a pixel with its coordinate x, y.

4.2 Preprocessing Hardware

Block diagram of the binary quantizer is shown in Fig. 5. The circuit performs eq. (1) and it picks up the red grove portion and binary quantizes it. The histogram is calculated by accumulating the pixel values with the same x or y coordinate. After the histogram calculation, the contents of the memories, i.e., histogram data is sequentially sent to the DFT unit.

Fig. 6(A) shows the 22-point DFT unit containing 22 TDFTS units, each of which does one-point DFT calculation, i.e., equations (3) ∼ (5). Fig. 6(B) shows the TDFTS unit which includes a direct digital frequency synthesizer (DDFS) that generates tcos() and tsin(), and the rest of the circuit performs the DFT calculation. The DDFS is made of an adder, register and phase-amplitude converter. Synchronized to the clock pulse, frequency control word k is accumulated in the register. Using the upper 3-bit of the register as a phase S, the phase-amplitude converter generates tsin(S) and tcos(S). Conventional DDFS uses read-only memory to generate more precise sin(S) or cos(S), but the proposed

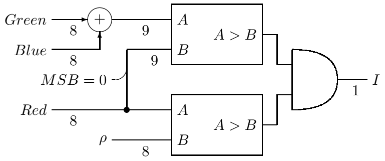


Fig. 5. Block diagram of the binary quantizer

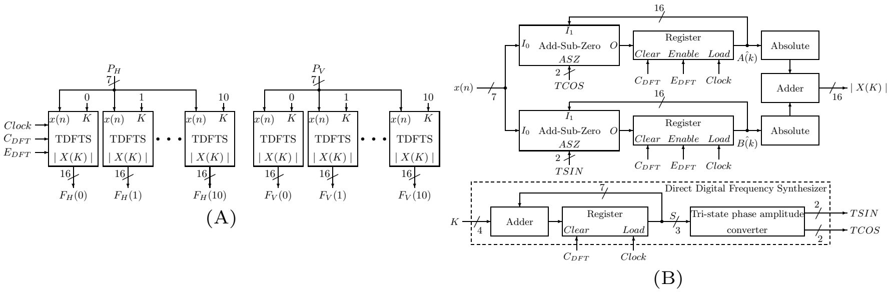


Fig. 6. DFT unit. (A) TDFT22 (22-point DFT), (B) TDFTS (a single point DFT).

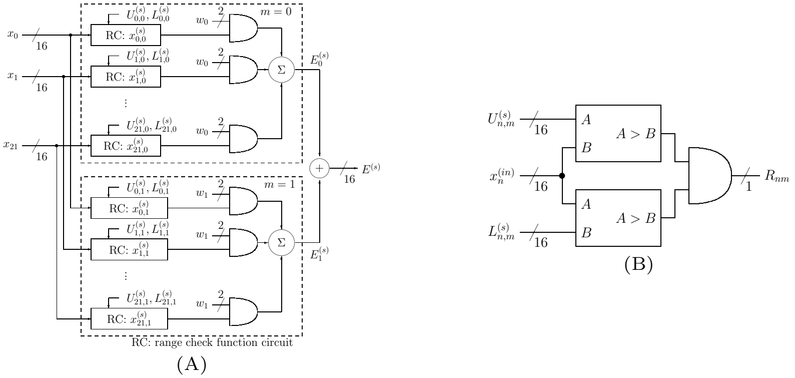


Fig. 7. Neuron. (A) Neuron unit, (B) Range check function circuit.

system uses combinatorial logic. Output frequency of the DDFS, f_{DDFS} is given by the following equation.

$$f_{DDFS} = \frac{K}{2^L} \cdot f_{CK} \tag{9}$$

L is the bit length of the register, f_{CK} is the frequency of the clock signal.

The calculations of $\hat{A}(k)$ and $\hat{B}(k)$ are carried out by the Add-Sub-Zero unit and register. The proposed DFT uses the Add-Sub-Zero circuit instead of the multiplier and it performs following operation.

$$O = \begin{cases} I_1 + I_0 & \text{if } ASZ = \text{“}01''\\ I_1 - I_0 & \text{if } ASZ = \text{“}11''\\ I_1 & \text{otherwise} \end{cases} \tag{10}$$

4.3 Classifier Network Hardware

The hardware neuron is shown in Fig. 7(A), which is made of range check function circuit and the circuit to calculate the weighted sum of the function values. The range check function circuit is depicted in Fig. 7(B). As the system

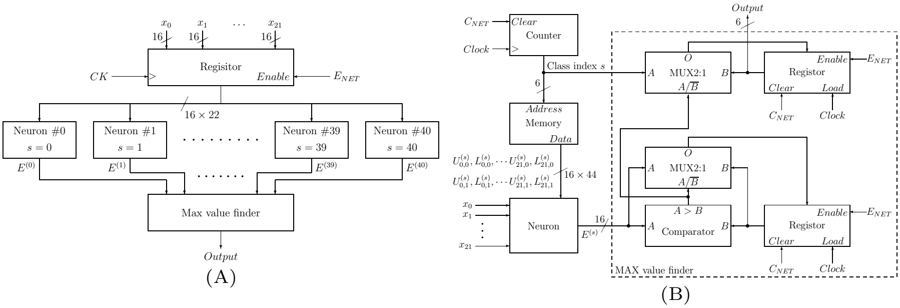


Fig. 8. Classifier network. (A) Parallel network, (B) Serial network.

is configured with $M = 2$, the neuron unit contains two circuits that calculate $E_0^{(s)}$ and $E_1^{(s)}$ and the sum of them is given as $E^{(s)}$ for the class s.

The classifier network can be implemented with either a parallel or a serial architecture. Fig.8(A) shows the parallel network. The network includes 41 neurons, each of which is assigned to different class and all neurons work in fully parallel. The input vector is fed to all neurons simultaneously, and their outputs are fed to max value find circuit to find the hand sign index s by searching for the neuron that gives the largest estimate value $E^{(s)}$.

The serial classifier network is made of a single neuron unit, a memory, a counter and a serial max value find circuit as shown in Fig. 8(B). The upper and lower limits data are loaded into the neuron from the memory, and the estimate values are obtained sequentially, thus the network requires 41 clocks to perform the recognition. The counter gives the memory address that is also treated as the class index s. In the max value find circuit, the new $E^{(s)}$ is compared to the reference value that is the largest one at that time. If the new one is larger than the reference, then the new $E^{(s)}$ and s are stored in the registers as the reference and its class index s.

4.4 Circuit Size and Speed

The two systems were designed by VHDL and the circuit sizes are estimated by logic synthesis. EDA tool from XILINX corporation is used to synthesize the logic. Before the logic synthesis, VHDL simulations were carried out to verify that the designs are correct. Specifically, all $F_H(n)$, $F_V(n)$, $E^{(s)}$ and classifier results are compared to those from C simulations, and it was verified that all of them are identical to the C simulation results.

Tab. 2 summarizes the circuit sizes of the systems. The recognition system with the serial network uses 90,000-gate while the system with the parallel network requires 280,000-gate equivalent hardware resources. The maximum frequencies of the clock signal for the systems are 52 MHz for the system with the serial network, and 24.5 MHz for the system with the parallel network.

Table 2. Circuit sizes

Circuit type	Gate count	Clock freq.
Recognition system with serial classifier	89,695	51.2 MHz
Recognition system with parallel classifier	282,341	24.5 MHz

5 Conclusions

This paper has described the hardware design of a hand sign recognition system with a simplified DFT. The new DFT uses heavily quantized version of sine/cosine functions. The use of the simplified DFT reduces the hardware cost of the system at a cost of slight degradation in performance. The simulation results shows that the degradation of the performance can be suppressed by training the classifier network with the images in various positions.

The hardware design of the recognition system was conducted to estimate the circuit size and operating speed. With the parallel classifier network, the circuit size of the recognition system is over 280,000 gate while the system with the serial classifier network requires about 90,000 gate of hardware resources. Regarding the operating speed, it has been revealed that the operation speed of the both system is fast enough to process NTSC video frame in real time. Thus the proposed hardware recognition system can be extended so that it can recognizes the video sequence, i.e., gesture recognition.

This work was supported by KAKENHI (19500153).

References

1. Pavlovic, V.I., Sharma, R., Huang, T.S.: Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review. IEEE Trans. Pattern Analysis and Machine Intelligence 19, 677–695 (1997)
2. Triesch, J., von der Malsburg, C.: A System for Person-Independent Hand Posture Recognition against Complex Backgrounds. IEEE Trans. Pattern Analysis and Machine Intelligence 23(12), 1449–1453 (2001)
3. Hoshino, K., Tanimoto, T.: Realtime Hand Posture Estimation with Self-Organizing Map for Stable Robot Control. IEICE Trans. on Information and Systems E89-D(6), 1813–1819 (2006)
4. Bobick, A.F., Wilson, A.D.: A state-based approach to the representation and recognition of gesture. IEEE Trans. Pattern Analysis and Machine Intelligence 19(12), 1325–1337 (1997)
5. Yang, H.-H., Ahuja, N., Tabb, M.: Extraction of 2D motion trajectories and its application to hand gesture recognition. IEEE Trans. Pattern Analysis and Machine Intelligence 24(8), 1061–1074 (2002)
6. Fujimura, H., Sakai, Y., Hikawa, H.: Japanese Hand Sign Recognition System. In: Ishikawa, M., Doya, K., Miyamoto, H., Yamakawa, T. (eds.) ICONIP 2007, Part I. LNCS, vol. 4984, pp. 983–992. Springer, Heidelberg (2008)
7. Hikawa, H., Fujimura, H.: Simplified DFT for Hand Posture Recognition System. In: Proc. NOLTA 2008, pp. 112–115 (September 2008)

Blind Source Separation System Using Stochastic Arithmetic on FPGA

Michihiro Hori and Michihito Ueda

Advanced Technology Research Laboratories, Panasonic Corporation
3-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0237, Japan
{hori.michihiro,ueda.michihito}@jp.panasonic.com

Abstract. We investigated the performance of a blind source separation (BSS) system based on stochastic computing in the case of an aperiodic source signal by both simulation and a field programmable gate array (FPGA) experiment. We confirmed that our BSS system can successfully infer source signals from mixed signals. We show that the system succeeds in separating source signals from mixed signals after about 3.7 seconds at a clock frequency of 32 MHz on an FPGA.

1 Introduction

The rapid development of very large scale integration (VLSI) technologies and the miniaturization of metal oxide semiconductor field effect transistors (MOSFETs) means for new challenges in noise filtering, which cannot be handled by conventional deterministic computing systems. In order to overcome such problems, stochastic computing (SC) systems, which are inspired by neurons, have been studied [1], [2], [3], [8]. These systems compute by utilizing pulse sequences related to analog quantities and offer various advantages including robustness in the presence of noise. Especially appealing properties of SC systems are the feasibility for simple circuitry and utilization of conventional digital elements. Therefore, SC systems have been applied to massive circuits such as artificial neural networks.

Recently, blind source separation (BSS) systems have become of interest in the field of soft computing. These systems can infer source signals from mixed signals when received by sensors. They are expected to apply to signal detection technologies in various fields like biomedical signal analysis. One particular method using a neural network has been proposed by Cichocki and Unbehauen [4]. This method succeeds in separating source signals from mixed signals even under even poor conditions. However, the synaptic weights incorporated can become arbitrary real numbers that are unpredictable. Consequently, if this method is implemented in hardware, there is the possibility that the synaptic weights fall outside the range over which hardware can operate correctly. Therefore, we previously proposed an improved method based on a SC system and investigated the case of periodic signals as source signals. We confirmed that this system could separate source signals from mixed signals on a field programmable gate array (FPGA) [6]. Here, we extend the previous study by investigating aperiodic source signals on an FPGA.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 843–850, 2009.

2 Blind Source Separation Using Stochastic Arithmetic

First, we describe the framework for BSS. We assume that the source signals s_j(t) (j = 1, 2, ..., n) are unknown, mutually independent and have zero-mean signals. The mixed signals observed by the sensors are x_i(t) (i = 1, 2,…, n). Moreover, for simplicity, we assume that the number of source signals is equal to the number of mixed signals, namely the number of sensors, and that there is a linear relation between the observed mixed signals and the source signals,

$$\boldsymbol{X}(t) = \boldsymbol{AS}(t) \quad , \tag{1}$$

where $\boldsymbol{X}$(t) = [x_1(t), x_2(t),…, x_n(t)]^T is the vector of mixed signals, $\boldsymbol{S}$(t) = [s_1(t), s_2(t),…, s_n(t)]^T is the vector of source signals, and $\boldsymbol{A}$ = [a_{ij}] is the mixing matrix in which a_{ij} are unknown real mixing parameters. We assume that $\boldsymbol{A}$ is nonsingular, namely det($\boldsymbol{A}$) ≠ 0. The system uses a neural network algorithm to output the inferred signals $\boldsymbol{Y}$ (t) = [y_1(t), y_2(t),…, y_n(t)]^T in real time. $\boldsymbol{Y}$ (t) can be of arbitrary amplitude and does not necessarily correlate to the index of the source signal. Therefore, the system succeeds in separating source signals from mixed signals if the output signals have the following combination of source signals,

$$\boldsymbol{Y}(t) = \boldsymbol{DQS}(t) \quad , \tag{2}$$

where $\boldsymbol{D}$ is a diagonal scaling matrix that represents the scales of amplitudes of inferred output source signals with nonzero entries, and $\boldsymbol{Q}$ is any permutation matrix in which the order of the inferred output source signals is taken into account.

Next, we explain the learning algorithm that Cichocki and Unbehauen proposed. They considered a single layer neural network consisting of n linear neurons as follows,

$$\boldsymbol{Y}(t) = \boldsymbol{W}(t)\boldsymbol{X}(t) \quad , \tag{3}$$

where $\boldsymbol{W}$ (t) = [w_{ij}] is the matrix of the adjusted synaptic weights. The learning algorithm that Cichocki and Unbehauen proposed was,

$$\frac{d\boldsymbol{W}(t)}{dt} = \mu(t)\left\{\boldsymbol{\Lambda} - \boldsymbol{f}\left[\boldsymbol{y}(t)\right]\boldsymbol{g}^T\left[\boldsymbol{y}(t)\right]\right\}\boldsymbol{W}(t) \quad , \tag{4}$$

$$\text{With } \boldsymbol{W}(0) \neq 0 \text{ and } \mathbf{det}\,\boldsymbol{W}(0) \neq 0 \quad ,$$

where μ(t) > 0 is the learning rate, and Λ = diag{λ_1, λ_2,…, λ_n} is a diagonal matrix with amplitude scaling factors (λ_i > 0). Moreover, $\boldsymbol{f}$[$\boldsymbol{y}$(t)] and $\boldsymbol{g}$[$\boldsymbol{y}$(t)] are the vectors of nonlinear functions of which parameters are the output signal, i.e., $\boldsymbol{f}$[$\boldsymbol{y}$(t)] = [f[y_1(t)], f[y_2(t)],…, f[y_n(t)]]^T and $\boldsymbol{g}$[$\boldsymbol{y}$(t)] = [g[y_1(t)], g[y_2(t)],…, g[y_n(t)]]^T. We can find a stationary solution using the learning algorithm if the following conditions are satisfied,

$$E\{f[y_i(t)]g[y_j(t)]\} = 0 \text{ for } i \neq j \,, \tag{5}$$

and

$$\lambda_i = E\{f[y_i(t)]g[y_i(t)]\} \text{ for } i = j \,. \tag{6}$$

However, synaptic weights in (4) can become unpredictable, arbitrary real numbers. During learning, the values of the synaptic weights can fall outside the designed range in which the circuits can process them correctly. This makes it difficult to implement the Cichocki and Unbehauen system in actual circuits. To resolve this problem, we proposed that synaptic weights and the amplitude scaling factors are multiplied by an updating weight, α ($0 < \alpha < 1$), if the values of the synaptic weights fall outside the designated range in the designed circuits [6]. The method is as follows. If one of the synaptic weights, w_{ij} (j = 1,2,…,n), in an output signal, y_i(t), falls outside the designed range, we multiply all w_{ij} (j = 1,2,…,n) in y_i(t) and the amplitude scaling factor, λ_i, by an updating weight, α, i.e.,

$$w_{ij} \Rightarrow \alpha w_{ij} \text{ and } \lambda_i \Rightarrow \alpha\lambda_i \quad (j = 1,2,\ldots,n)\,. \tag{7}$$

Moreover, updating y_i(t) is independent of updating y_j(t) (i ≠ j). As a result, the system can separate source signals from mixed signals with the synaptic weights remaining in the designed range. It is our expectation that this system can be applied to mobile electric devices like a hearing aid.

We confirmed that our system can separate source signals from mixed signals including only periodic source signals when using appropriate synaptic weights on an FPGA [6]. In this paper, we extend that previous work by investigating aperiodic source signals on an FPGA. The block diagram of the circuit and algorithm are shown in Fig. 1 and 2, respectively. In Fig. 1, mixed signals x_1(t) and x_2(t), both of which are digital signals, are converted into pulse sequences by stream generators, which are SC elements that convert a digital quantity into a stochastic pulse sequence [7]. These pulse sequences are then individually multiplied by the pulse sequences of the synaptic weights using SC multipliers and are added by SC adders. The system outputs the pulse sequences of the inferred signals, y_1(t) and y_2(t). Using these pulse sequences, the synaptic weights are adjusted according to the learning algorithm we proposed based on SC. In Fig. 2, we show the block diagram of the learning algorithm for the

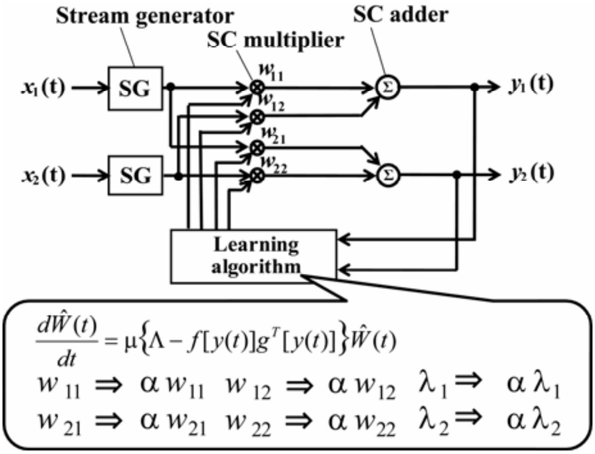

Stream generator
SC multiplier
SC adder
x_1(t)
x_2(t)
SG
SG
w_{11}
w_{12}
w_{21}
w_{22}
Σ
Σ
y_1(t)
y_2(t)
Learning algorithm
$\frac{d\hat{W}(t)}{dt} = \mu\{\Lambda - f[y(t)]g^T[y(t)]\}\hat{W}(t)$
$w_{11} \Rightarrow \alpha w_{11}$ $w_{12} \Rightarrow \alpha w_{12}$ $\lambda_1 \Rightarrow \alpha\lambda_1$
$w_{21} \Rightarrow \alpha w_{21}$ $w_{22} \Rightarrow \alpha w_{22}$ $\lambda_2 \Rightarrow \alpha\lambda_2$

Fig. 1. Block diagram of a circuit for BSS based on a SC system

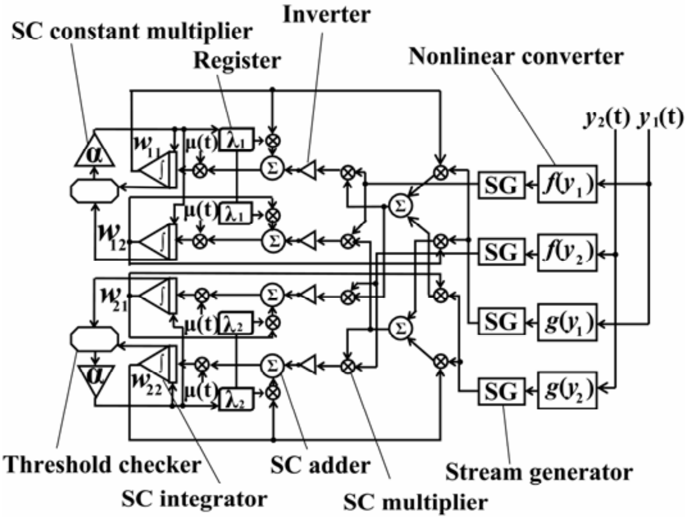


Fig. 2. The learning algorithm of a BSS circuit based on the SC system in Fig. 1

circuit in Fig. 1. The pulse sequence y_1(t) and y_2(t) is injected into nonlinear converters, which convert the input signals into output pulses according to nonlinear function elements that consist of a counter and a register storing the nonlinear functions. Stream generators then convert the output of the nonlinear converters into pulse sequences. We then arrange SC multipliers, SC adders, registers which store the amplitude scaling factors and a learning rate, and inverters in a way to satisfy eqn. 4. Finally, SC integrators output the pulse sequences with synaptic weights w_{11}, w_{12}, w_{21}, and w_{22}. A SC integrator is constructed by an up-down counter. It can execute time integration by counting pulses. It should be noted that the bit accuracy of up-down counters in SC integrators is related to the convergence of the circuit. Next, we configure threshold checkers and SC constant multipliers. When any synaptic weights reach the maximum or minimum of the up-down Counters in the SC integrators, the SC constant multipliers appropriately multiply an updating weight by the amplitude scaling factor and the values of the up-down Counters in the SC integrators. As a result, the synaptic weights are able to converge and remain in the designed range and can separate source signals from mixed signals.

Next, we show the simulation results of our SC system based BSS circuit for an aperiodic source signal. We assumed the system had two sources for simplicity. Therefore, the system had two inputs and two outputs. The source signals were,

$$\begin{aligned} s_1 &= \mathbf{sin}(2\pi \times \frac{k}{1024}) \times \mathbf{sin}(2\pi \times 30 \times \frac{k}{1024}), \\ s_2 &= \mathrm{UR(uniform\ random\ signal)}\ (\text{-}1 \le \mathrm{UR} \le 1) \end{aligned} \tag{8}$$

,

where k is the time step, the mixing matrix is

$$\boldsymbol{A} = \begin{pmatrix} 0.2 & 0.8 \\ 0.5 & 0.5 \end{pmatrix}. \tag{9}$$

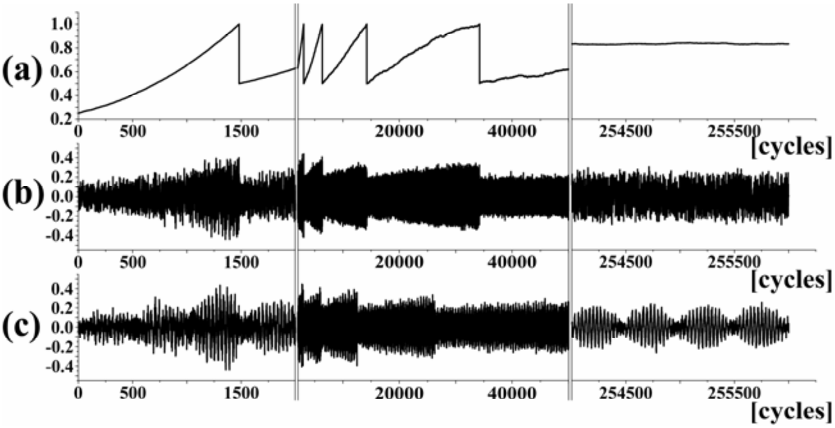


Fig. 3. Simulation results of the learning process for the algorithm in the case of an aperiodic source signal. (a) synaptic weight w_{11}, (b) waveform of output signal y_1(t), (c) waveform of output signal y_2(t).

Moreover, we assumed $f[y(t)] = y(t)^2\text{sign}(y(t))$ and $g[y(t)] = \tanh(10\, y(t))$. To simplify the circuit implementation, we assumed $\mu(t) = 1$. Fig. 3(a) represents the synaptic weight w_{11}, and Fig. 3(b) and (c) are waveforms for the inferred signals y_1(t) and y_2(t), respectively, during learning. We can see that our system can separate source signals from mixed signals when updating the synaptic weights. Then, we investigated the convergence of our system using the commonly used performance index E_1 described in ref. [5] as follows,

$$E_1 = \sum_{i=1}^{n}\left(\sum_{j=1}^{n}\frac{|p_{ij}|}{\mathbf{max}_k |p_{ik}|}-1\right)+\sum_{j=1}^{n}\left(\sum_{i=1}^{n}\frac{|p_{ij}|}{\mathbf{max}_k |p_{kj}|}-1\right) \qquad (10)$$

,

where p_{ij} is the product of the matrix of synaptic weights and the mixing matrix as follows,

$$\boldsymbol{P} = [p_{ij}] \equiv \boldsymbol{W}(t)\boldsymbol{A} \quad . \qquad (11)$$

If the system separates the source signals from mixed signals completely, E_1 is zero, which is its minimum value. In Fig. 4, we show the simulation result for average E_1 after 50 trials with bit accuracy for the SC integrators in the circuit. When the bit accuracy of the SC integrators in the circuit is less than 17 bits, the value of E_1 is unstable. On the other hand, the number of cycles at the convergence point of E_1 is large when the bit accuracy of the SC integrators in the circuit is larger than 19 bits. Therefore, we chose the bit accuracy of the SC integrators to be 18bits. We can see that the number of cycles at the convergence point reaches about 200,000 for 18bits accuracy. We confirmed that our SC circuit can separate source signals from mixed signals even when the mixed signals include an aperiodic source signal.

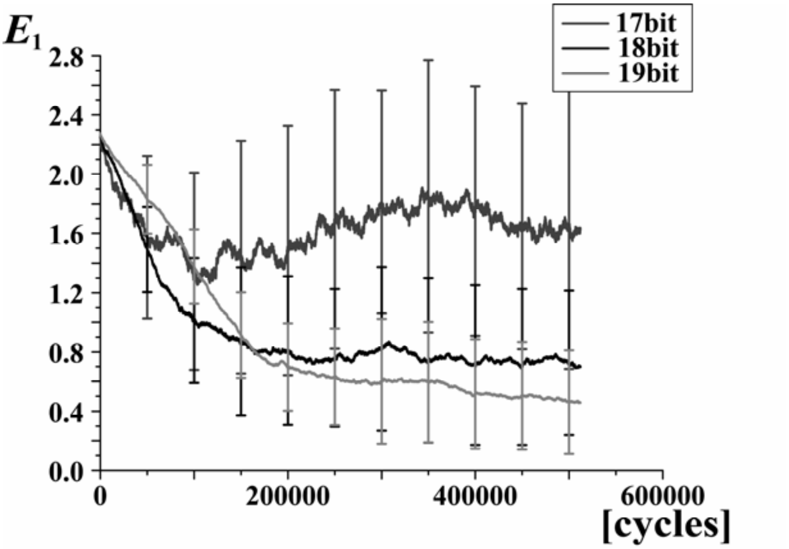


Fig. 4. The dependence of the performance index E_1 during learning in a SC system with bit accuracy for the SC integrators. The line represents the average E_1 after executing 50 trials. Error bars show standard deviation.

3 FPGA Implementation

We implemented our circuit onto an FPGA board. The design has been described in Verilog-HDL. The FPGA device model employed was a Xilinx Spartan 3 (XC3S400), the clock frequency was 32 MHz, the rate injecting the digital values of the mixed signals into our circuit was 15.6 kHz, and the bit accuracy of the Digital-to-Analog Converter (DAC) was 10 bits. The digital values of the mixed signals were 7bits, SC Integrators were 18bits, registers storing the amplitude scaling factors were 18bits,

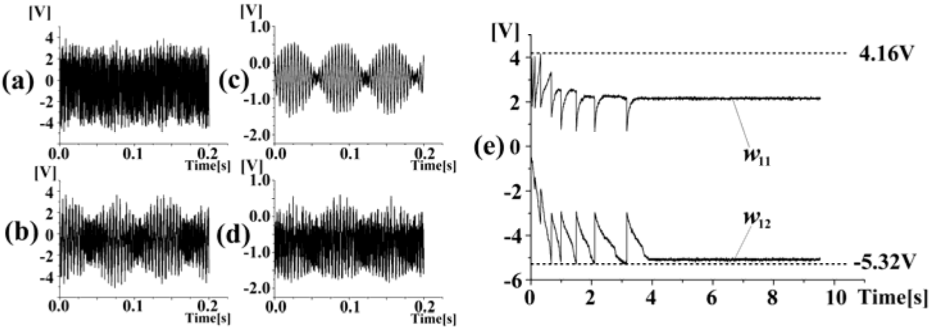


Fig. 5. FPGA experimental results (a) input signal x_1(t), (b) input signal x_2(t), (c) output signal y_1(t), (d) output signal y_2(t), (e) synaptic weights w_{11} and w_{12} during learning

nonlinear converters were 10bits, and the updating weight was 0.5. Fig. 5(a) and (b) show the input signals, namely mixed signals. In Fig. 5(e), we represent the synaptic weights w_{11} and w_{12} during learning. The maximum and minimum voltages of the designed range were 4.16 V and -5.32 V, respectively. We can see that the synaptic weights can converge with updating on the case that the voltages of synaptic weights reach the maximum or minimum, and are within the desired boundary. The waveform of the output signals after learning are shown in Fig. 5(c) and (d). Finally, the performance index E_1 of our circuit from FPGA experiment is seen in Fig. 6. E_1 converges at about 3.7 seconds. We confirmed our system operates correctly in an FPGA board even for an aperiodic source.

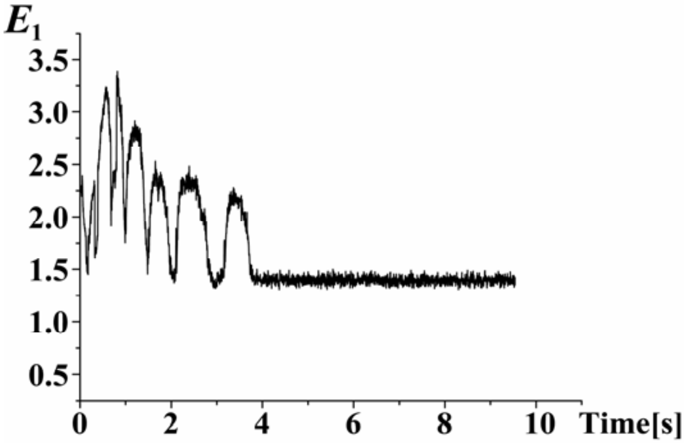


Fig. 6. FPGA results of the performance index E_1 during learning

4 Conclusions

We investigated the performance of a BSS system for the case of an aperiodic source signal through simulations and on an FPGA board. We found that synaptic weights converge at about 200,000 cycles and that our system can separate source signals from mixed signals even when they include an aperiodic source signal. We also confirmed that the circuit we proposed can operate correctly on an FPGA board and that the performance index E_1 converges at about 3.7 seconds at a clock frequency of 32 MHz.

References

1. Gains, B.R.: Stochastic Computing Systems. In: Tou, J.F. (ed.) Advances in Information Systems Science, ch. 2, vol. 2, pp. 37–172. Plenum, New York (1969)
2. Hori, M., Ueda, M., Iwata, A.: Stochastic Computing Chip for Measurement of Manhattan Distance. Japanese Journal of Applied Physics 45(4B), 3301–3306 (2006)

3. Ueda, M., Yamashita, I., Morita, K., Setsune, K.: Stochastic Associative Processor Operated by Random Voltages. IEICE Trans. Electron E90-C(5), 1027–1034 (2007)
4. Cichocki, A., Unbehauen, R.: Robust Neural Networks with On-Line Learning for Blind Identification and Blind Separation of Sources. IEEE Trans. Circuits and Systems-I 43(11), 894–906 (1996)
5. Amari, S., Cichocki, A., Yang, H.H.: A New Learning Algorithm for Blind Signal Separation. In: Touretzky, D., Mozer, M., Hasselmo, M. (eds.) Advances in Neural Information Processing Systems, vol. 8, pp. 757–763. MIT Press, Cambridge (1996)
6. Hori, M., Ueda, M.: FPGA Implementation of a Blind Source Separation System based on Stochastic Computing. In: 2008 IEEE Conference on Soft Computing in Industrial Applications, pp. 182–187. IEEE Press, New York (2008)
7. Daalen, M., Jeavons, P., Shawe-Taylor, J., Cohen, D.: A Device for Generating Binary Sequences for Stochastic Computing. Electronic Letters 29(1), 80–81 (1993)
8. Köllmann, K., Riemschneider, K.R., Zeidler, H.C.: On-Chip Back-propagation Training Using Parallel Stochastic Bit Streams. In: Proc. 5th Intern. Conf. on Microelectronics for Neural Networks and Fuzzy Systems, pp. 149–156 (1996)

Noise-Tolerant Analog Circuits for Sensory Segmentation Based on Symmetric STDP Learning

Gessyca Maria Tovar, Tetsuya Asai, and Yoshihito Amemiya

Graduate School of Information Science and Technology, Hokkaido University
Kita 14, Nishi 9, Kita-ku, Sapporo, 060-0814 Japan
gessyca@sapiens-ei.eng.hokudai.ac.jp
http://lalsie.ist.hokudai.ac.jp/

Abstract. We previously proposed a neural segmentation model suitable for implementation with complementary metal-oxide-semiconductor (CMOS) circuits. The model consists of neural oscillators mutually coupled through synaptic connections. The learning is governed by a symmetric spike-timing-dependent plasticity (STDP). Here we demonstrate and evaluate the circuit operation of the proposed model with a network consisting of six oscillators. Moreover, we explore the effects of mismatch in the threshold voltage of transistors, and demonstrate that the network was tolerant to mismatch (noise).

1 Introduction

One of the most challenging problems in sensory information processing is the analysis and understanding of natural scenes, i.e., images, sounds, etc. These scenes can be decomposed into coherent "*segments*". The segments correspond to different components of the scene. Although this ability, generally known as sensory segmentation, is performed by the brain with apparent ease, the problem remains unsolved. Several models that perform segmentation have been proposed [1]-[3], but they are often difficult to implement in practical integrated circuits. In [4] we proposed a simple neural segmentation model that is suitable for analog CMOS circuits. The model consisted of mutually-coupled neural oscillators. The oscillators were coupled with each other through positive or negative synaptic connections.

In this paper, we demonstrate and evaluate the circuit operation of the proposed model with a network consisting of six oscillators. Moreover, we conduct Monte-Carlo simulations to study the effects of threshold mismatch among transistors in our network using three oscillators, and we demonstrate that the network is tolerant to the mismatch (noise).

2 The Model

Our segmentation model is shown in Fig. 1(a). The network has N Wilson-Cowan type neural oscillators (u_i output of i-th activator and v_i output of i-th

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 851–858, 2009.

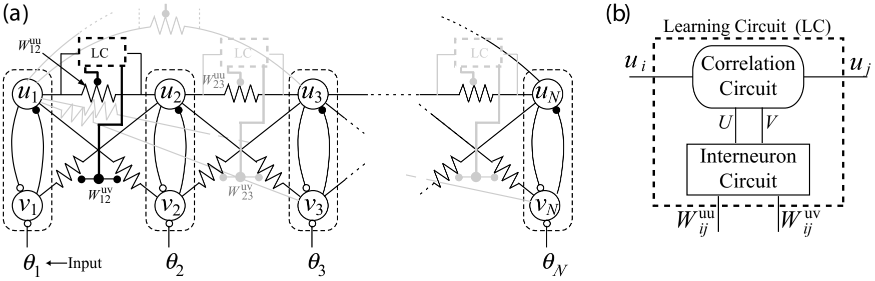


Fig. 1. a) Network construction of segmentation model, and b) learning circuit model

inhibitor). The oscillators are coupled with each other through resistive synaptic connections. The dynamics are defined by

$$\tau \frac{du_i}{dt} = -u_i + f_{\beta_1}(u_i - v_i) + \sum_{j \neq i}^{N} W_{ij}^{\mathrm{uu}} u_j, \tag{1}$$

$$\frac{dv_i}{dt} = -v_i + f_{\beta_2}(u_i - \theta_i) + \sum_{j \neq i}^{N} W_{ij}^{\mathrm{uv}} u_j, \tag{2}$$

where τ represents the time constant, N is the number of oscillators, θ_i is the external input to the i-th oscillator, and $f_{\beta_i}(x)$ is the sigmoid function defined by $f_{\beta_i}(x) = [1 + \tanh(\beta_i x)]/2$. Also, W_{ij}^{uu} represents the connection strength between the i-th activator and j-th activator, and W_{ij}^{uv} the strength between the i-th activator and the j-th inhibitor. Each neuron accepts external inputs, e.g., sound inputs, and oscillates (or does not oscillate) when the input amplitude is higher (or lower) that a given threshold. For a more detailed explanation, refer to [4].

The easiest way to segment neurons is to connect the activators belonging to the same (or different) group with positive (or negative) synaptic weights. However, circuits that implement positive and negative weights may occupy a large area on analog LSIs, which prevents us from implementing large-scale networks. Therefore, instead of using negative weights, we used positive synaptic weights between the activator and inhibitors [4]. These weights are updated by learning circuits. As shown in Fig 1(a), the learning circuits (LCs) are located between two activators. Each of them consists of a correlation circuit and an interneuron circuit (see Fig. 1(b)). Therefore, let us start with the explanation of the correlation circuit. In our model, neurons should be correlated (or anti-correlated) if they receive synchronous (or asynchronous) inputs. Based on this assumption, the synaptic weights are updated on the basis of symmetric spike-timing-dependent plasticity (STDP) using Reichardt's correlation neural network [5]. The basic unit is illustrated in Fig. 2(a). It consists of a delay neuron (D) and a correlator (C). The delay neuron produces blurred (delayed) output D_{out} from spikes produced by activator (u_i). The dynamics are given

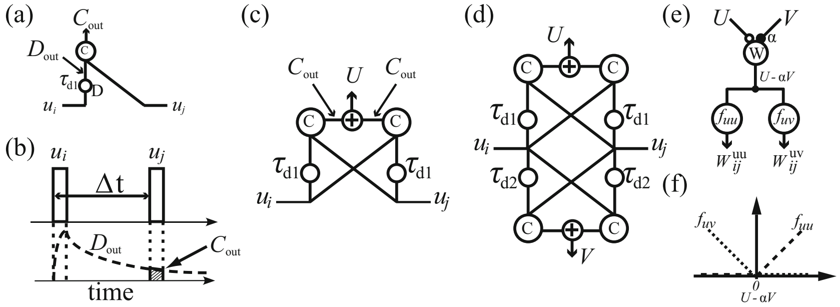


Fig. 2. a) Basic unit of Reichardt's correlation network, b) Reicherdt's network operation, c) unit pair, d) correlation circuit, e) model of interneuron circuit, and f) input-output characteristics of piecewise linear functions (f_{uu} and f_{uv})

by $\tau_{\rm d1} dD_{\rm out}/dt = -D_{\rm out} + u_i$, where $\tau_{\rm d1}$ represents the delay time constant. The correlator accepts $D_{\rm out}$ and spikes produced by activator (u_j), and outputs $C_{\rm out}$ ($\equiv D_{\rm out} \times u_j$). The operation is illustrated in Fig. 2(b). Since this basic unit can calculate correlation values only for positive inter-spike intervals Δt, we used a unit pair consisting of two basic units, as shown in Fig. 2(c). The output (U) is obtained by summing the two $C_{\rm out}$s. Through temporal integration of U, we obtained a Gaussian-type response for Δt [4]. The sharpness of this response increased as $\tau_{\rm d1} \rightarrow 0$, so introducing two unit pairs with different time constants, e.g., $\tau_{\rm d1}$ and $\tau_{\rm d2}$ ($\tau_{\rm d1} \ll \tau_{\rm d2}$), we obtained two responses for U and V with different sharpness. Then, the weighted subtraction ($U - \alpha V$) produced the well-known Mexican-hat characteristic that we used as STDP in the oscillator network [4]. The correlation circuit is shown in Fig. 2(d).

The two outputs (U and V) of the correlation circuits are given to the interneuron circuit shown in Fig. 2(e). Interneuron W receives outputs of the correlation circuit (U and V), and performs the weighted subtraction ($U - \alpha V$). When $U - \alpha V$ is positive, neurons u_i and u_j in Fig. 2(d) should be correlated, and the weight between activators ($W_{ij}^{\rm uu}$) should be increased. On the other hand, when $U - \alpha V$ is negative, the neurons should be anti-correlated, and the weight between the activator and inhibitor ($W_{ij}^{\rm uv}$) should be increased. The output of interneuron W is given to two additional interneurons (f_{uu} and f_{uv}). The input-output characteristics of these interneurons are shown in Fig. 2(f). The outputs of these interneurons are given to the weight circuit (represented by resistors in the model; Fig. 1(a)) in order to modify the positive resistances. For a more detailed explanation and simulation of the model refer to [4].

We newly carried out numerical simulations to evaluate the "segmentation ability," which represents the number of survived segments after the learning. The number of segments as a result of the network's learning strongly depends on the STDP characteristic as well as the input timing of neurons (Δt). Let us remember that neurons that fire "simultaneously" should be correlated. "Simultaneously" is to be defined by some "time windows of coincidence" that we call $\sigma_{\rm STDP}$. Thus, neurons that receive inputs within the time windows should be

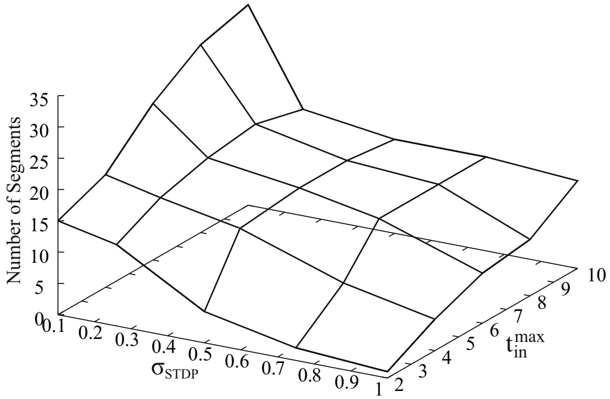


Fig. 3. Simulation results showing segmentation ability of the network

correlated. Simulation results are shown in Fig. 3. The number of neurons (N) was set to 50. The neurons received random inputs within time $t_{\mathrm{in}}^{\max}$ (maximum input timing). We observed that when σ_{STDP} was 1 and neurons received their inputs within time 2, the number of segments was about 2. The contrary was observed when σ_{STDP} was 0.1 and $t_{\mathrm{in}}^{\max}$ was 10, where the number of segments was about 35.

3 CMOS Circuits

Construction of a single neural oscillator is shown in Fig. 4(a). The oscillator consists of two standard differential amplifiers (a differential pair and a current mirror) and two additional capacitors C_1 and C_2. A circuit implementing Reichardt's basic unit (see Fig. 2(a)) is shown in Fig. 4(b). The circuit has a delayer

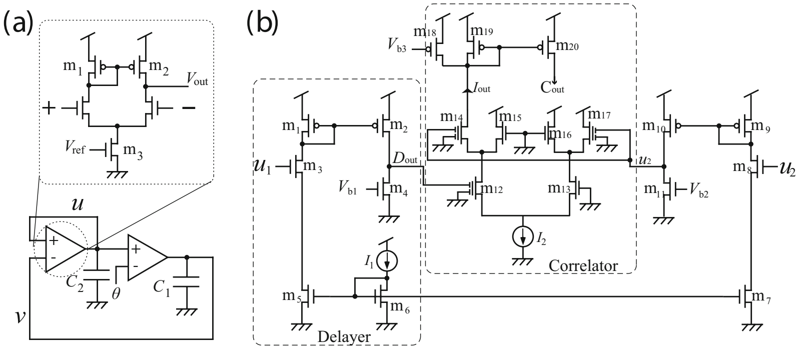


Fig. 4. a) Neural oscillator circuit [6], and b) Reichardt's basic unit circuit [6]

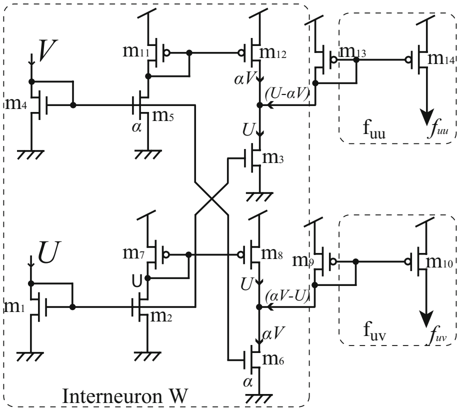


Fig. 5. Interneuron circuit

and a correlator. The delayer consists of a bias current source (I_1), current mirrors (m_1-m_2 and m_5-m_6) and a pMOS source-common amplifier (m_2-m_4). The correlator consists of three differential pairs (m_{12}-m_{13}, m_{14}-m_{15} and m_{16}-m_{17}), a pMOS current mirror (m_{19}-m_{20}), a bias transistor (m_{18}) and a bias current source (I_2). We employed floating gate MOS FETs for m_{12}, m_{14} and m_{17} to decrease the gain of the differential pairs. Detailed operations and simulation results of these two circuits are explained in [6].

A basic circuit implementing the interneurons (W, f_{uu} and f_{uv}) is shown in Fig. 5. The circuit consists only of current mirrors. Input current U (from Reichardt's circuit; correlation circuit) is copied to m_3 by current mirror m_1-m_3, and is copied to m_8 by current mirrors m_1-m_2 and m_7-m_8. At the same time, input current V is copied to m_6 by current mirror m_4-m_6, and is copied to m_{12} by current mirrors m_4-m_5 and m_{11}-m_{12}. Recall that we need the subtraction of $U - \alpha V$ to produce the Mexican-hat characteristic. Therefore, we set the weight (α) as $\alpha \equiv W_5/L_5 \cdot L_4/W_4 = W_6/L_6 \cdot L_4/W_4$, where W_i and L_i represent the channel width and length of transistor m_i, respectively. So, when current U is higher than current αV, current f_{uu} is outputted by current mirror m_{13}-m_{14}. Otherwise, current f_{uv} is outputted by current mirror m_{11}-m_{12}.

4 Simulation Results

First we carried out circuit simulations for the interneuron circuit. The parameters used for the transistors were obtained from MOSIS AMIS 1.5-μm CMOS process. Transistors sizes (W/L) were 4 μm/1.6 μm for m_1-m_4, 10 μm/1.6 μm for m_5 and m_6, 4.5 μm/16 μm for m_8 and m_{12}, 3.5 μm/16 μm for m_{13}, and 4 μm/16 μm for the rest transistors. The supply voltage was set to 5 V. Input current V was set to 100 nA, and input current U varied from 0 to 200 nA. The simulation results are shown in Fig. 6. When $U + \Delta I < V$ where $\Delta I \approx 20$ nA, output current f_{uv} flowed and f_{uu} was 0. When $U - V < \Delta I$, both f_{uu} and f_{uv} were 0. When hen $U - \Delta I > V$, f_{uu} flowed while f_{uv} remained at 0.

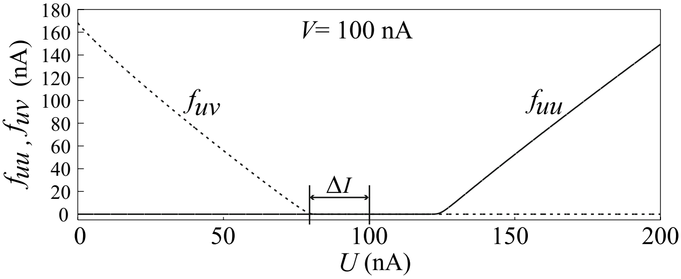


Fig. 6. Circuit simulation results of interneuron circuit

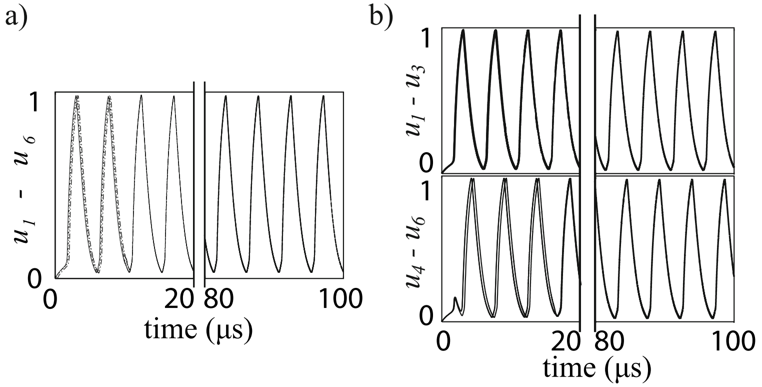


Fig. 7. Circuit simulation results for a) inter-spike interval $\Delta t = 0$, and b) $\Delta t = 3$ μs

Next, we carried out circuit simulations of the circuit network with $N = 6$. Transistor sizes (W/L) for the Recichardt's basic circuit (see Fig. 4(b)) were 4 μm/1.6 μm for nMOS transistors and m_{20}, and 4 μm/16 μm for the rest of the transistors. Voltages V_{b2} and V_{b3} were set to 550 mV and 4.08 V respectively, while V_{b1} was set to 510 mV for delay τ_{d1}, and was set to 430 mV for delay τ_{d2}. With these settings, we obtained positive W ($U - \alpha V$; see Fig. 2(e)) for $|\Delta t| \leq 1$ μs, and obtained negative W for $|\Delta t| > 1$ μs. In other words, when $|\Delta t| \leq 1$ μs, neurons should be correlated, otherwise, they should be anti-correlated, as explained before.

The normalized time courses of u_is ($i = 1 \sim 6$) are shown in Figs. 7(a) and (b). As shown in Fig. 7(a), at $t = 0$, external inputs θ_i ($i = 1 \sim 6$) were 2.5 V, which is equivalent to Δt=0. We observed that all neurons were gradually synchronized. On the contrary, Fig. 7(b) shows that at $t = 0$ external inputs $\theta_{1,2,3}$ were set to 2.5 V, and inputs $\theta_{4,5,6}$ were set to 0. Then, at $t = 3$ μs $\theta_{4,5,6}$ were set to 2.5 V, which is equivalent to $\Delta t = 3$ μs. We observed that $u_{1,2,3}$ and $u_{4,5,6}$ were desynchronized without breaking synchronization among neurons in the same group that were gradually synchronized. This indicated that segmentation of neurons based on the input timing was successfully achieved.

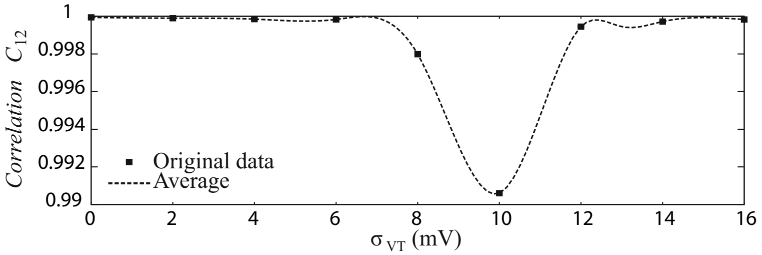


Fig. 8. Correlation values between neurons u_1 and u_2 for different σ_{VT}

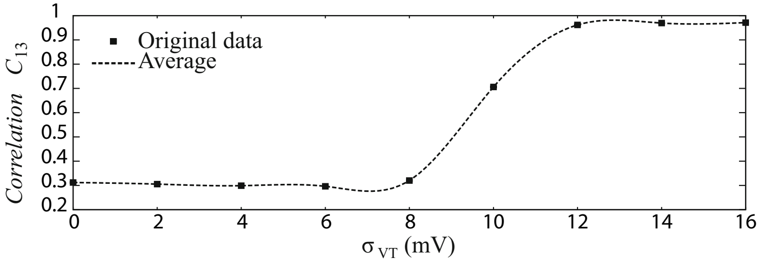


Fig. 9. Correlation values between neurons u_1 and u_3 for different σ_{VT}

To consider the noise tolerance of the network, we carried out Monte-Carlo simulations in our circuit network with $N = 3$. The parameter V_{th} (threshold voltage) of all transistors was varied using Gaussian noises with standard deviation σ_{VT}. When $t = 0$, external inputs to neurons (θ_1, θ_2, θ_3) were set to (2.5,0,0)V. Then, at $t = 1$ μs, (θ_1, θ_2, θ_3) were set to (2.5,2.5,0)V, whereas they were set to (2.5,2.5,2.5)V at $t = 2.4$ μs. In other words, neurons u_1 and u_2 should be synchronous with each other, and they should be asynchronous with u_3 because of Δt=1.4 μs. To evaluate the performance of the network, we calculated correlation values C_{ij} between neurons u_i and u_j given by

$$C_{ij} = \frac{\langle u_i u_j \rangle - \langle u_i \rangle \langle u_j \rangle}{\sqrt{\langle u_i^2 \rangle - \langle u_i \rangle^2}\sqrt{\langle u_j^2 \rangle - \langle u_j \rangle^2}}. \tag{3}$$

We calculated C_{12} and C_{13} to evaluate the synchronicity between segments. Figures 8 and 9 show the simulation results. As observed in the figures, when σ_{VT} <10 mV neurons u_1 and u_2 were correlated, while the correlation value (C_{13}) between neurons u_1 and u_3 was low, i.e., they were anti-correlated. Due to imperfections of the CMOS fabrication process, device parameters, e.g., threshold voltage, etc., suffer large variations [7]. These variations among transistors cause a significant change in general analog circuits. Nevertheless, the results obtained in Figs. 8 and 9 showed that our network successfully segmented neurons for σ_{VT}s lower than 10 mV, which indicated that the network is tolerant to threshold mismatch among transistors.

5 Conclusion

Previously, we proposed a neural segmentation model that is suitable for analog VLSIs using conventional CMOS technology. We proposed a novel segmentation method based on a symmetric spike-timing dependent plasticity (STDP) using Reichard's correlation neural networks. In this paper, we evaluated the segmentation ability of the network through numerical simulations. In addition we proposed and evaluated basic circuits for constructing segmentation hardware. We demonstrated the operation of the circuit network using six neurons. Finally, we explored the effect of threshold mismatches among transistors in our network with three oscillators, and showed that the network was tolerant to device mismatches.

References

1. Han, S.K., Kim, W.S., Kook, H.: Temporal segmentation of the stochastic oscillator neural network. Physical Review E 58, 2325–2334 (1998)
2. Von der Malsburg, C., Schneider, W.: A neural cocktail-party processor. Biological Cybernetics 54, 29–40 (1986)
3. Wang, D.L., Terman, D.: Locally excitatory globally inhibitory oscillator networks. IEEE Trans. on Neural Networks. 6(1), 283–286 (1995)
4. Fukuda, E.S., Tovar, G.M., Asai, T., Hirose, T., Amemiya, Y.: Neuromorphic CMOS Circuits Implementing a Novel Neural Segmentation Model Based on Symmetric STDP Learning. J. Signal. Proc. 11(6), 439–444 (2007)
5. Reichardt, W.: Principles of Sensory Communication. Wiley, New York (1961)
6. Tovar, G.M., Fukuda, E.S., Asai, T., Hirose, T., Amemiya, Y.: Analog CMOS Circuits Implementing Neural Segmentation Model Based on Symmetric STDP Learning. In: 14th International conference on Neural information Processing, Japan, pp. 306–315 (2007)
7. Pelgrom, M.J.M., Duinmaijer, A.C.J., Welbers, A.P.G.: Matching Properties of MOS Transistors. J. Solid-State Circuits 24(5), 1433–1440 (1989)

A Novel Approach for Hardware Based Sound Classification

Mauricio Kugler, Victor Alberto Parcianello Benso,
Susumu Kuroyanagi, and Akira Iwata

Department of Computer Science and Engineering
Nagoya Institute of Technology
Showa-ku, Gokiso-cho, 466-8555, Nagoya, Japan
mauricio@kugler.com, benso@mars.elcom.nitech.ac.jp,
{bw,iwata}@nitech.ac.jp

Abstract. Several applications would emerge from the development of efficient and robust sound classification systems able to identify the nature of non-speech sound sources. This paper proposes a novel approach that combines a simple feature generation procedure, a supervised learning process and fewer parameters in order to obtain an efficient sound classification system solution in hardware. The system is based on the signal processing modules of a previously proposed sound processing system, which convert the input signal in spike trains. The feature generation method creates simple binary features vectors, used as the training data of a standard LVQ neural network. An output temporal layer uses the time information of the sound signals in order to eliminate the misclassifications of the classifier. The result is a robust, hardware friendly model for sound classification, presenting high accuracy for the eight sound source signals used on the experiments, while requiring small FPGA logic and memory resources.

1 Introduction

By the information provided from the hearing system, the human being can identify any kind of sound (sound recognition) and where it comes from (sound localization) [1]. If this ability could be reproduced by artificial devices, many applications would emerge, from support devices for people with hearing loss to safety devices.

In contrast to sound localization, systems capable of identifying the nature of non-speech sound sources were not deeply explored. Some authors study the application of speech-recognition techniques [2], while others attempt to divide all possibly mixed sound sources and apply independent techniques for each kind of signal [3]. Sakaguchi, Kuroyanagi and Iwata [4] proposed a sound classification system based on the human auditory model, using spiking neural networks for identifying six different sound sources.

Due to its high computational cost, sound classification systems are often implemented in hardware for real-time applications. A preliminary hardware

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 859–866, 2009.

implementation of the model proposed in [4] was presented in [5], while a full system implementation, including the signal processing modules, was proposed in [6]. The spiking neurons based model proposed in [4,5], although presenting acceptable results, requires the adjustment of several critical parameters, while requiring a large FPGA area, in spite of claims of implementation efficiency of spiking neural networks in digital hardware.

This paper proposes a new approach for sound classification and its correspondent hardware implementation. While still based on spikes, a new feature generation method enables high accuracy with an efficient implementation in hardware. The proposed method also presents few non-critical parameters on the learning process.

The organization of the paper goes as follows: a short description of the signal processing and pulse generation modules is presented in Section 2, and Section 3 introduces the proposed model, which hardware implementation is presented in Section 4. Section 5 presents experimental results, and Section 6 concludes the paper with analysis of the results and suggests possible future extensions.

2 Signal Preprocessing and Spikes Generation

The main structure of the sound classification system is shown in Figure 1(a), with its first block is detailed in Figure 1(b). The use of spikes is required due to the use of the proposed method in a larger system, described previously in [6], which also includes sound localization and orientation detection, in which sharing of signal processing modules is mandatory.

The sound signal is sampled at 48kHz, converted to single-precision floating-point representation and sent to the filter bank module, which divides it in N frequency channels. After, the signals' envelops are extracted and their amplitude used for controlling the period of the spikes generators. All spike trains $p_n(t)$ $(n = 1 \ldots N)$ become the input data of the sound classification module.

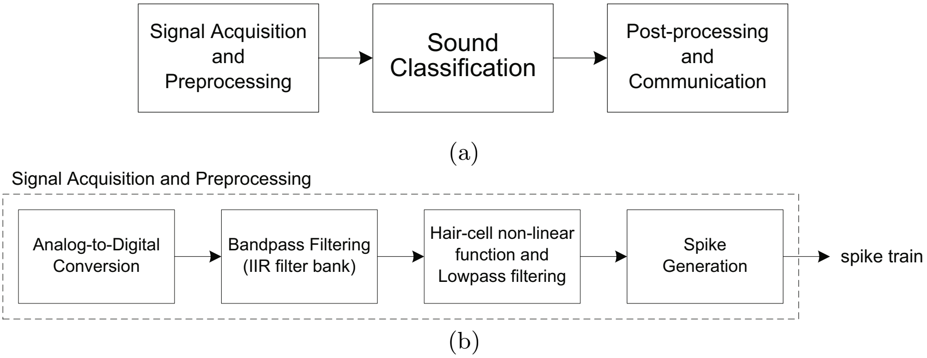

Signal Acquisition and Preprocessing
Sound Classification
Post-processing and Communication
(a)
Signal Acquisition and Preprocessing
Analog-to-Digital Conversion
Bandpass Filtering (IIR filter bank)
Hair-cell non-linear function and Lowpass filtering
Spike Generation
spike train
(b)

Fig. 1. Sound classification system (a) main structure and (b) signal processing

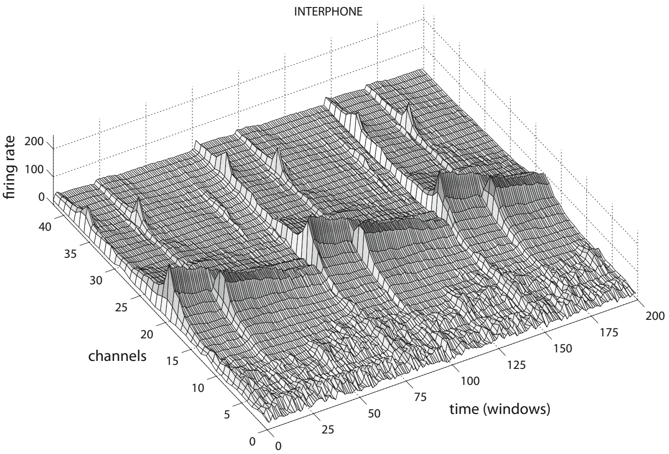


Fig. 2. Spike map for a segment of the *interphone* sound signals

3 Proposed Model

The previous sound classification model proposed in [4,5] presents several parameters that must be correctly tuned in order to obtain a good accuracy. The tuning process is not straightforward and requires several retrials. Moreover, as the model is based on a non-supervised learning model, a successful learning cannot be always guaranteed and the training process is very time-consuming.

The approach proposed in this paper combines a simple feature generation procedure, a supervised learning process and fewer parameters in order to obtain an efficient sound classification system solution. This solution presents high accuracy and uses small resources (e.g. FPGA area and memory). The following sections present each of the modules in details.

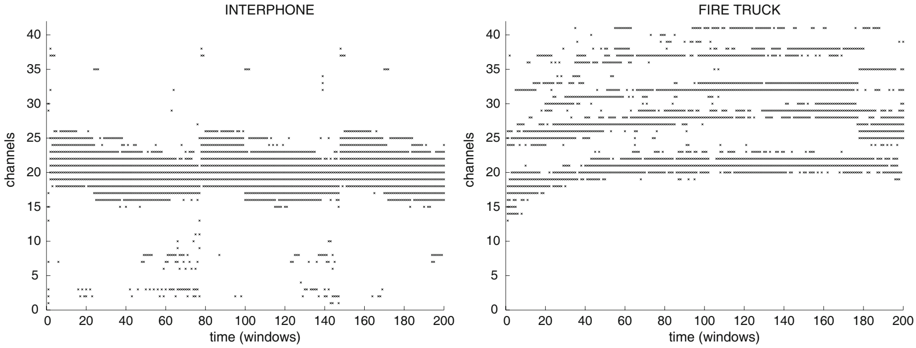
3.1 Feature Generation

As the firing rate of the spike trains corresponds to the amplitude of the signal, a straightforward approach for detecting the energy $x_n(t)$ of each n^{th} frequency channel is to count the number of spikes in a time window of length W:

$$x_n(t) = \sum_{i=0}^{W-1} p_n(t-i) \tag{1}$$

where $p_n(t)$ is the spike train value on time t, $n = 1 \ldots N$. The vector $\mathbf{x}$ hence represents the sound pattern in time t. Figure 2 shows the plot of the number of spikes per time window for the sound signal *interphone*, used on the experiments in Section 5.

If this vector is naively used as the input sample $\mathbf{z}$ for the classifier, different signal amplitudes would result in different patterns, what is not desirable. A



(a)

(b)

Fig. 3. Binary features for the (a) *interphone* and (b) *fire truck* sound signals

different approach is to consider the highest firing rate channels' indexes as the feature vector for the classifier:

$$z_m = \arg\max_n{}^m p_n \tag{2}$$

where $\arg\max^i$ represents the index of the i^{th} highest element in a vector, $m = 1 \dots M$ and M is the number of features (number of channels to be considered). The drawback of this approach is that, due to the use of the channels' indexes, small frequency shifts result in large vectorial distances. Therefore, standard distance measurements cannot be applied for comparing the patterns.

Even though the order of the highest firing rates may contain information about the pattern, a very robust set of features can be obtained by ignoring this information. Thus, the new feature vector becomes a binary vector defined as:

$$z_n = \begin{cases} 1 & if \quad p_n \geq \max_i{}^M p_i \\ 0 & otherwise \end{cases} \tag{3}$$

where $\max^i$ represents the i^{th} highest element in a vector. Figure 3 shows the binary features of the *interphone* and *fire truck* sound signals.

3.2 Classification

The standard Learning Vector Quantization (LVQ) neural network [7] was used as the classifier. The learning rate α was reduced linearly along the training epochs by a β factor. The clusters centers were initialized using the Max-Min Distance clustering algorithm [8].

As the patterns were reduced to simple binary vectors, they can be compared by Hamming distance:

$$d(\mathbf{z}, \omega) = \sum_{i=1}^{N} |z_i - \omega_i| \tag{4}$$

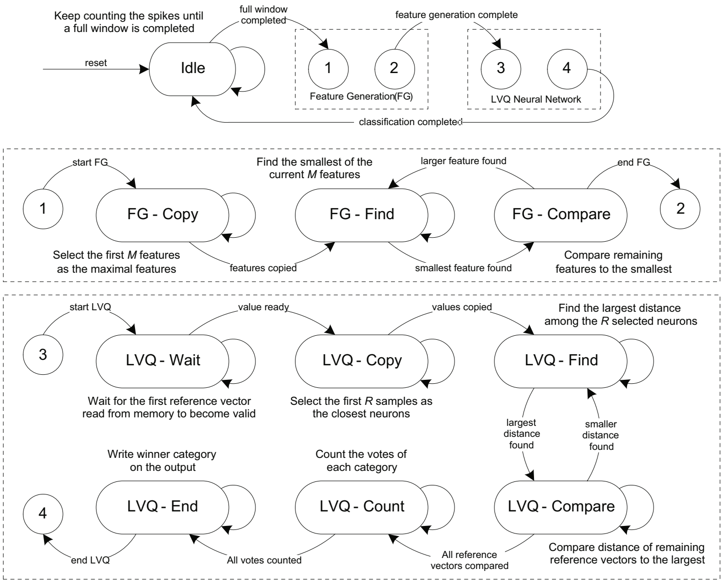

Keep counting the spikes until a full window is completed
full window completed
feature generation complete
reset
Idle
1
2
3
4
Feature Generation(FG)
LVQ Neural Network
classification completed
start FG
Find the smallest of the current M features
larger feature found
end FG
FG - Copy
FG - Find
FG - Compare
Select the first M features as the maximal features
features copied
smallest feature found
Compare remaining features to the smallest
start LVQ
value ready
values copied
Find the largest distance among the R selected neurons
LVQ - Wait
LVQ - Copy
LVQ - Find
Wait for the first reference vector read from memory to become valid
Select the first R samples as the closest neurons
largest distance found
smaller distance found
Write winner category on the output
Count the votes of each category
LVQ - End
LVQ - Count
LVQ - Compare
end LVQ
All votes counted
All reference vectors compared
Compare distance of remaining reference vectors to the largest

Fig. 4. Sound classification system's state machines

where the elements of sample **z** and weight ω, during the training process, are converted to binary values only for distance calculation.

3.3 Time Potentials

In the feature generation process, no feature extraction or selection procedure is being used in order to "clean" the patterns. Although such techniques would avoid misclassifications by the LVQ neural network, they also would add more parameters to the system and increase the training complexity.

Up to the LVQ neural network, no time information had been used for the classification. It can be assumed that the sound sources being recognized will not present instant changes, i.e. they last for periods of time much larger than the size of the time windows. Thus, by the use of potentials similar to the membrane potential of spiking neurons, one can remove the instant errors from the LVQ neural network without modifying the training process. The time potentials are defined as:

$$u_k(t) = \begin{cases} \min\left(u_{\max}, u_k(t-1) + \gamma\right) & if \quad k = y(t) \\ \max\left(0, u_k(t-1) - 1\right) & if \quad k \neq y(t) \end{cases} \tag{5}$$

where u_k is the potential of the k^{th} category, γ is the increment for the winner category and $u_{\max}$ is the maximal potential. Hence, the winner category at time t is the one with higher $u_k(t)$ value. It must be noted that, by setting the $u_{\max}$ parameter, the increment γ does not need to be adjusted. In the experiments of this paper, γ was set fix to 2.

4 Hardware Implementation

The state machine of the sound classification system is shown in Figure 4. The spike counting happens as an independent process, which transfer the total number of spikes to the feature generation process when a full window is completed. The feature generation module sequentially searches the M largest spike rates and send this result to the classification module, which searches for the cluster(s) with the smallest distance to the feature vector. Finally, a winner-takes-all voting is performed and the final label is sent to the output.

The circuit was implemented in an Altera Stratix II EPS260F484C4, which contains 48352 Adaptive Look-Up Tables (ALUT) and more than 2M bits of internal memory. The proposed method (with 9 features, 1000 clusters per class and a 1000 samples time window) uses a total of 2136 ALUTs, 1206 dedicated logic registers (DLR) and 184K bits of memory (for storing the reference vectors of the LVQ neural network). The size of the memory scales linearly with the number of clusters per class and the number of categories, while the number of ALUTs and DLRs presents small increases for larger values of R and number of categories. The number of clusters per class and number of features do not significantly increase the number of ALUTs and DLRs.

For this same configuration, the average processing time for the feature extraction and classification procedures are, respectively, is 3.74μs and 42.16μs. The classification processing time grows linearly with the total number of clusters. Considering a 48kHz sampling rate, these timings enable the use of a much larger number of categories and reference vectors.

5 Experiments

Eight sound signals were used on the experiments: *alarm bell*, *ambulance*, *fire truck*, *interphone*, *kettle*, *phone ring*, *police car* and *human voice*. The databases where split in training and test sets in a 2:1 rate. The *voice* database contains several individuals' voice signals, thus, presenting a larger number of samples.

The LVQ network was trained with $\alpha_0 = 0.1$, $\beta = 0.995$ and a maximal of 1000 learning epochs. Figure 5 shows the test accuracy for several training

Table 1. LVQ neural network confusion matrix

Original	Recognition Result								
	alarm	*ambulance*	*fire truck*	*interphone*	*kettle*	*phone*	*police*	*voice*	*unknown*
alarm	**1512**	0	0	1	13	0	0	2	0
ambulance	0	**3543**	1	870	0	0	18	1	91
fire truck	42	6	**4349**	7	16	28	96	0	40
interphone	0	121	6	**1985**	0	0	4	0	22
kettle	40	0	0	0	**1479**	0	0	0	1
phone	1	0	1	0	0	**1765**	0	0	1
police	11	106	147	565	13	46	**4039**	0	181
voice	59	902	202	1850	1	70	62	**6931**	387

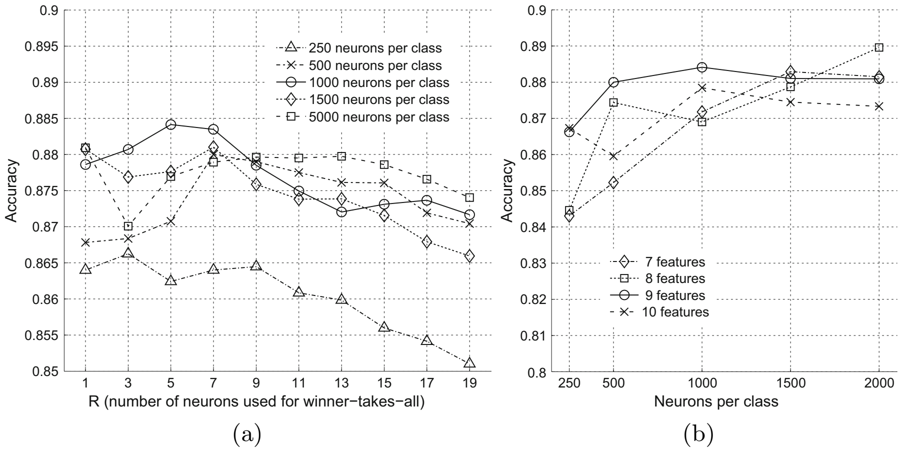


Fig. 5. LVQ accuracy (a) as function of R and (b) as function of the number of features

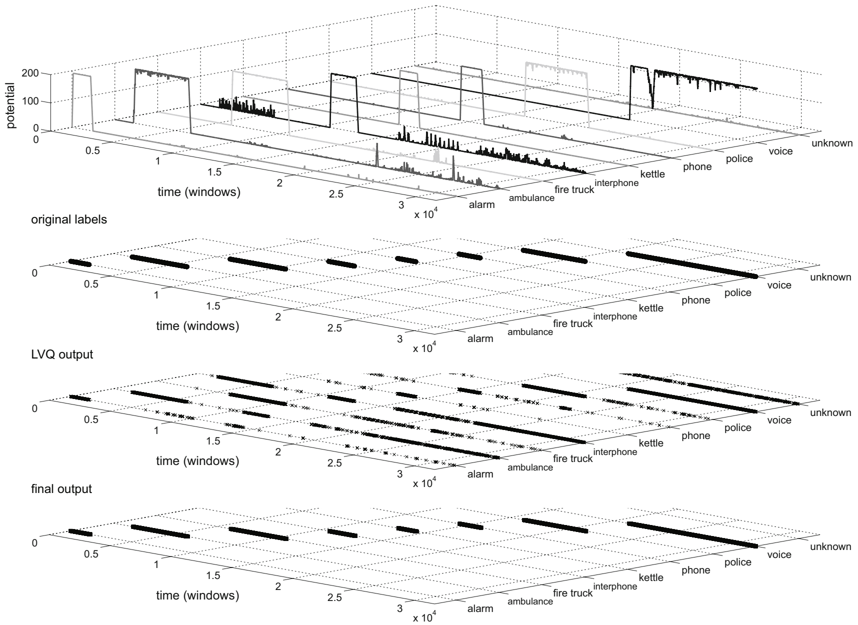


Fig. 6. Time potentials and final sound classification results

parameters, and Table 1 shows the confusion matrix of the chosen parameters set (9 features, 1000 neurons per class and 1000 samples time window).

The accuracy values presented in Figure 5 and Table 1 are the raw classification result of the LVQ neural network. Figure 6 show the results when calculating the time potentials, for a maximal potential u_{max} equal to 192. All the misclassifications were eliminated, with the the drawback of a small delay introduced on the response of the system.

6 Discussion and Conclusions

This paper proposed a new method for implementing a sound recognition system in an FPGA device. A novel and robust feature generation approach permits the use of a very simple classifier, a standard LVQ neural network, while an independent temporal layer eliminate the misclassifications.

The obtained classification accuracy is encouraging. When running in the FPGA, the proposed model showed to be very robust and insensitive to background noise. On the case of the *human voice* database, several individual's voice not used on the training set were successfully recognized. An improved version of the time potential layer with a better response time is being developed.

Future works include the use of a larger filter bank in order to increase the system's accuracy. Several more sound sources will be used in order to determine the limits os accuracy of the proposed system. The implementation of the learning system in hardware would permit several new applications, as well as increasing the system's flexibility.

References

1. Pickles, J.O.: An Introduction to the Physiology of Hearing, 2nd edn. Academic Press, London (1988)
2. Cowling, M., Sitte, R., Wysocki, T.: Analysis of speech recognition techniques for use in a non-speech sound recognition system. In: Proceedings of the 6th International Symposium on Digital Signal Processing for Communication System, Manly, TITR, January 2002, pp. 16–20 (2002)
3. Turk, O., Sayli, O., Dutagaci, H., Arslan, L.M.: A sound source classification system based on subband processing. In: Hansen, J.H.L., Pellom, B. (eds.) Proceedings of the 7th International Conference on Spoken Language Processing, Denver, September 2002, pp. 641–644 (2002)
4. Sakaguchi, S., Kuroyanagi, S., Iwata, A.: Sound discrimination system for environment acquisition. Technical Report NC99-70, Nagoya Institute of Technology, pp. 61–68 (December 1999)
5. Iwasa, K., Kugler, M., Kuroyanagi, S., Iwata, A.: A sound localization and recognition system using pulsed neural networks on FPGA. In: Proceedings of the 20th International Joint Conference on Neural Networks, Orlando, August 2007, pp. 1252–1257. IEEE Computer Society, Los Alamitos (2007)
6. Kugler, M., Iwasa, K., Benso, V.A.P., Kuroyanagi, S., Iwata, A.: A complete hardware implementation of an integrated sound localization and classification system based on spiking neural networks. In: Ishikawa, M., Doya, K., Miyamoto, H., Yamakawa, T. (eds.) ICONIP 2007, Part II. LNCS, vol. 4985, pp. 577–587. Springer, Heidelberg (2008)
7. Fausett, L.: Fundamentals of Neural Networks: architectures, algorithms and applications. In: Neural Networks Based on Competition, 1st edn., pp. 156–217. Fundamentals of Neural Networks, New Jersey (1994)
8. Friedman, M., Kandel, A.: Introduction to Pattern Recognition: statistical, structural and fuzzy logic approaches. In: Classification by Distance Functions and Clustering, 1st edn., pp. 73–77. Imperial College Press, London (1999)

The Generalized Product Neuron Model in Complex Domain

B.K. Tripathi*, B. Chandra, and P.K. Kalra

Indian Institute of Technology,
Kanpur - 208016, India
{abkt,bchandra,kalra}@iitk.ac.in

Abstract. This paper proposes a complex valued generalized product neuron (GPN) which tries to incorporate polynomial structure in the aggregation of inputs. The advantage of using this model is to bring in the non-linearity in aggregation function by taking a product of linear terms in each dimension of the input space. This aggregation function has the ability to capture higher-order correlations in the input data. Such neurons are capable of learning any problem irrespective of whether the multi dimensional data is linearly separable or not which resembles higher order neurons. But these neurons do not have combinatorial increase of the number of weights in the dimensions of inputs as higher order neurons. The learning and generalization capabilities of proposed neuron are demonstrated through variety of problems. It has been shown that some benchmark problems can be solved with single GPN only without hidden layer.

1 Introduction

The spatial integration of synapses on the dendritic tree reflects the computation performed by a neuron. Over the years, a substantial body of evidence has grown to support the non-linear integration of signals in biological neuron cells [1]. The well known artificial neuron models that provide higher order correlation among input signals are higher order neurons [2] and polynomial neurons [10]. All these structures have increased the computational power but they suffer from combinatorial increase in the number of weights. Further pi-sigma network was proposed in [14] which needs smaller number of weights as compared to other higher order neurons but much more than conventional summation neurons. This paper explores a neuron model which incorporates the higher order correlation among input components but the number of weights needed are same as in summation neurons. The primary motivation is to develop a higher order neuron model which will serve as a basic unit for multi layer network. Multiplicative operation being the most basic of all non-linearities, is the natural choice to include non-linearity in neural networks at the time of aggregation. Michel

* Corresponding author, Asstt. Professor, Department of Computer Science and Engineering, H B Technological Institute, Kanpur (India), Ph: 91-9415153771.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 867–876, 2009.

Schmitt study [8] shows that this operation can be used to implement second and higher order polynomial relation among set of inputs. In this paper we propose a non-linear aggregation function which is the product of linear functions in different dimensions of the input space. This function can represent simple polynomial relation among set of inputs. Thus, the proposed generalized product neuron (GPN) produces sum of all possible products of the weighted inputs in aggregation process. The superior convergence speed, reduction in learning parameters and ability to learn two dimensional motion naturally in complex valued neural network [11,12] have motivated to implement proposed neuron in complex domain. The empirical evaluation of this model has been carried out as single unit as well as in network with other conventional neurons. It has proved to be more efficient than other neuron models in real and complex domain.

The rest of the paper is organized as follows : section 2 introduces the structure of generalized product neuron, section 3 presents the learning rules based on complex back-propagation algorithm, section 4 examines the capability of new structure in function approximation, classification and functional mapping problems. Finally we discuss conclusions and ideas for future work in section 5.

2 Generalized Product Neuron (GPN)

In various neurons in human nervous system, neuro-biologists have also observed multiplicative way of aggregation. Various authors in past [3,9] have discussed the relevance of product operations, its power and advantage over traditional summing operation. Product operation captures second and higher order relation among set of inputs and for neurons to respond strongly to correlation among input pairs or groups one must include product terms in the model which one intends to develop [8]. The product operation proposed here for aggregation process is not simple multiplication of weighted inputs but it performs multiplication of linear terms in each dimension of the input space, as

$$\prod_{l=1}^{L} (w_{lm}\, z_l \;+\; 1) \tag{1}$$

Where $w_{lm}\, z_l \in \mathbb{C}$. The $\mathbb{C}$ is the set of complex numbers. w_{lm} is the weight that connects l^{th} neuron to m^{th} neuron and z_l is output of l^{th} neuron. Thus, the generalized product operation captures polynomial relation among set of input signals and yields the sum of all possible products of weighted inputs. The generalized product operation in Eq.(1) can also be expressed as -

$$\prod_{l=0}^{n}{}^{\oplus}\; w_{lm} z_l \tag{2}$$

$$= \mathit{Sum\ of\ the\ products\ of\ all\ combinations\ of\ weighted\ inputs}$$

$$= w_{0m} + w_{1m}z_1 + w_{2m}z_2 + \;\dots\; + w_{1m}z_1 \times w_{2m}z_2 + w_{1m}z_1 \times w_{3m}z_3 +$$

$$\dots\; + w_{1m}z_1 \times w_{2m}z_2 \times w_{3m}z_3 + \;\dots + w_{1m}z_1 \times w_{2m}z_2 \times \;\dots\; \times w_{Lm}z_L \tag{3}$$

The number of learning parameters in GPN are same as that of summation neuron which are significantly less than that of other higher order neurons but GPN constructs higher order terms in aggregation function by considering the simple combinations of all the weighted inputs. Thus proposed structure form a polynomial of inputs, Eq.(3), by performing multiplication operation over linear functions of different inputs Eq.(1). The order of polynomial depends on the number of inputs to the neuron. In this paper, the network of only summation neurons with non-linear activation function is referred as multilayered perceptron (MLP) and equivalent complex valued network is referred as $\mathbb{C}$-MLP.

3 Learning Rule

In a multi layer network of the *generalized product neuron in complex domain* ($\mathbb{C}$-GPN), all synaptic weights, bias and input-output signals are complex number. The activation function ($f_{\mathbb{C}}$) for $\mathbb{C}$-GPN is defined as 2-D extension of real activation function (f), as in [5,11]. Let $V = \Re(V) + j \times \Im(V)$[1] be the net potential of a neuron then output of the neuron is defined as :

$$y = f_{\mathbb{C}}(V) = f(\Re(V)) + j \times f(\Im(V)) \tag{4}$$

In complex theory, Liouville's theorem states that if a function $f_{\mathbb{C}}$ is regular at all $z \in \mathbb{C}$ and bounded then $f_{\mathbb{C}}$ is a constant function. Hence we have to select either regularity or boundedness. Expression (4) is non regular because the Cauchy-Reimann equation does not hold for this, so boundedness is chosen. Note that $-1 < \Re\left[f_{\mathbb{C}}\right], \Im\left[f_{\mathbb{C}}\right] < 1$. The simulations in this study are performed either with single $\mathbb{C}$-GPN or with a three layered network (L-M-N), taking M proposed neurons in hidden layer and N summation neurons in output layer. The learning rules for this $\mathbb{C}$-GPN network is derived using gradient descent method by minimizing mean square error (MSE) :

$$E = \frac{1}{2PN} \sum_{p=1}^{P} \sum_{n=1}^{N} |e_n^p|^2 \tag{5}$$

$e_n^p \in \mathbb{C}$ is the difference between actual and desired output of n^{th} neuron for p^{th} pattern in output layer. P is the number of patterns in training set. Let $\eta \in [0,1]$ is learning rate, $f^{'}$ is derivative of function f and $\overline{z}$ is complex conjugate of z. The complex back-propagation algorithm ($\mathbb{C}$-BP) minimizes cost function E by recursively altering the weight coefficients based on gradient descent, as -

$$w^{new} = w^{old} - \eta \bigtriangledown_w E \tag{6}$$

[1] $\Re$ and $\Im$ stands for real and imaginary components of complex value. j is imaginary unity.

Where the gradient $\bigtriangledown_w E$ is derived with respect to both real and imaginary parts of complex weights.

$$\begin{aligned}\varDelta w &= -\eta \ \bigtriangledown_w E = -\eta \ \bigtriangledown_{\Re(w)} E - j \ \eta \ \bigtriangledown_{\Im(w)} E \\ &= -\eta \left(\frac{\partial E}{\partial \Re(w)} + j \times \frac{\partial E}{\partial \Im(w)} \right)\end{aligned} \tag{7}$$

Let V_n is net potential of n^{th} output neuron $(n = 1..N)$ and Y_m be the output of m^{th} neuron $(m = 1..M)$ in hidden layer. For any weight $w = w_{mn}$ in output layer, from Eq.(7)

$$\varDelta w_{mn} = \frac{\eta}{N} \overline{Y}_m \left(\Re(e_n) \ f'(\Re(V_n)) + j \times \ \Im(e_n) \ f'(\Im(V_n)) \right) \tag{8}$$

Let net potential in a generalized product neuron at hidden layer is V_m^π and $l, k, h = 1...L$. This net potential can be expressed term wise from Eq. (2, 3)

$$V_m^\pi = w_{0m} z_0 + \sum_{l=1}^{L} w_{lm} z_l + \sum_{\substack{l,k=1 \\ l \neq k}}^{L} w_{lm} z_l \times w_{km} z_k + \sum_{\substack{l,k,h=1 \\ l \neq k \neq h}}^{L} w_{lm} z_l \times w_{km} z_k \times w_{hm} z_h + ... \tag{9}$$

For $w = w_{lm}$, the gradient of cost function (E) can be obtained by following the chain rule of derivation in Eq.(7) :

$$\varDelta w_{lm} = \frac{\eta}{N} \left(\Re(\varGamma_m^\pi) + j \times \Im(\varGamma_m^\pi) \right) \left(\frac{\partial(\Re(V_m^\pi))}{\partial \Re(w_{lm})} - j \times \frac{\partial(\Im(V_m^\pi))}{\partial \Re(w_{lm})} \right) \tag{10}$$

Where

$$\Re(\varGamma_m^\pi) = f'(\Re(V_m^\pi)) \sum_{n=1}^{N} \{ \Re(e_n) f'(\Re(V_n)) \Re(w_{mn}) + \Im(e_n) f'(\Im(V_n)) \Im(w_{mn}) \} \tag{11}$$

$$\Im(\varGamma_m^\pi) = f'(\Im(V_m^\pi)) \sum_{n=1}^{N} \{ \Im(e_n) f'(\Im(V_n)) \Re(w_{mn}) - \Re(e_n) f'(\Re(V_n)) \Im(w_{mn}) \} \tag{12}$$

$$\varDelta w_{lm} = \frac{\eta}{N} \ \overline{z_l} \ \ \varGamma_m^\pi \ \left(1 + \sum_{\substack{k=1 \\ k \neq l}}^{L} \overline{w_{km} z_k} + \sum_{\substack{k,h=1 \\ k \neq h \neq l}}^{L} \overline{w_{km} z_k} \times \overline{w_{hm} z_h} + ... \right) \tag{13}$$

4 Results and Discussion

4.1 Performance Evaluation with Single Neuron System

All the real domain problems discussed in this section are encoded for complex domain by assigning the input-output data to real part and zero to imaginary part of complex data, except detection of symmetry which has different encoding scheme.

Table 1. Simulation results for detection of symmetry problem

S.No.	Network	Neuron Type	Algorithm	Success rate	Average Epochs	Target Error	Parameters
1	3-2-2	Real MLP	Real-BP	89%	2484	0.1	14
2	3-1-1	$\mathbb{C}$-MLP	$\mathbb{C}$-BP	75%	326	0.1	12
3	3-1-1	$\mathbb{C}$-MLP	Improved $\mathbb{C}$-BP	89%	190	0.1	12
4	3-4-2	Real MLP	Real-BP	100%	1491	0.1	21
5	3-2-1	$\mathbb{C}$-MLP	$\mathbb{C}$-BP	100%	268	0.1	22
6	3-2-1	$\mathbb{C}$-MLP	Improved $\mathbb{C}$-BP	100%	130	0.1	22
7	*3-1*	$\mathbb{C}$-*GPN*	$\mathbb{C}$-BP	*100%*	*188*	*0.01*	*08*

The Detection of Symmetry. The real domain input-output mapping of this popular benchmark problem has been encoded for complex domain setup in [13]. This problem is solved in [13,4] with different three layered networks using conventional and improved back-propagation algorithms. The proposed single neuron is able to solve this encoded problem [13] without any hidden layer using conventional $\mathbb{C}$-BP. Table 1 presents the comparative analysis of results described in [13] and with $\mathbb{C}$-GPN. The results bring out the facts that proposed neuron is most efficient with least number of parameters.

The Iris Data Problem. The Fisher's Iris flower dataset is a well known 3-class problem. The classes are flower types and every pattern has four features of flower. The training was performed with 75 data points and testing on another 75 data points. The performance results using different networks are presented in Table 2. Results show that only one $\mathbb{C}$-GPN is able to achieve test error of 2.7% with least average epochs and learning parameters.

Box Jenkins Gas Furnace Dataset. In Box Jenkins gas furnace air and methane were combined in order to get a mixture of gases containing CO_2. In this dataset [6], gas flow rate x(t) is input and the CO_2 concentration is furnace output y(t). The furnace output y(t + 1) is modeled as a function of the previous output y(t) and input x(t-3). Half of the dataset is used for training and rest half for testing. The performance of single proposed neuron for this dataset is presented in Table 3 and Fig.(1). The Single generalized product neuron in complex domain is far efficient than other three layered network.

Table 2. Training and testing performance for Iris dataset

S.No.	Network	Neuron Type	Average Epochs	MSE	Parameters	Testing
1	4-4-1	Real-MLP	20000	0.0056	25	2.7%
2	4-2-1	$\mathbb{C}$-MLP	15000	0.0033	26	2.7%
3	*4-1*	$\mathbb{C}$-*GPN*	*7000*	0.0035	*10*	*2.7%*

Table 3. Performance comparison with Box Jenkins gas furnace dataset

S. No.	Network	Neuron Type	Parameters	MSE (Training)	MSE (Testing)	Epochs
1	2-2-1	Real-MLP	9	8.2×10^{-3}	2.2×10^{-2}	5000
2	2-1-1	$\mathbb{C}$-MLP	10	4.02×10^{-4}	2.3×10^{-3}	2000
3	2-1	$\mathbb{C}$-GPN	6	5.65×10^{-4}	1.91×10^{-3}	500

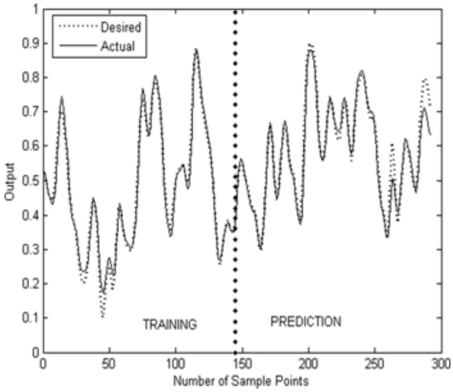


Fig. 1. Performance of $\mathbb{C}$-GPN for Box Jenkins Gas Furnace dataset

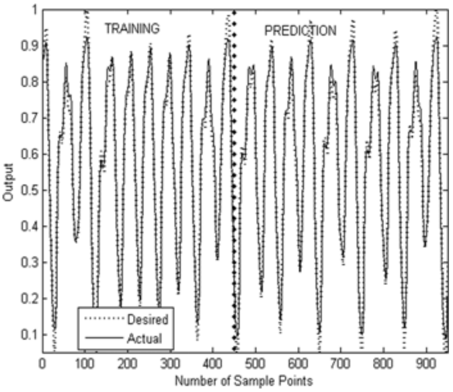


Fig. 2. Performance of $\mathbb{C}$-GPN for Mackey Glass time series

Mackey Glass time series. The Mackey Glass time series [7] is widely used for testing the performance of neuron models. This series presents a model for white blood cell production in leukemia patients and also has nonlinear oscillations. This is a chaotic time series which makes it an universally acceptable representation of nonlinear oscillations of many physiological processes. The MG delay difference equation is given as -

$$y(t+1) = (1-b)\, y(t) + a\, \frac{y(1-\tau)}{1+y^{10}(t-\tau)} \tag{14}$$

where $a = 0.2, b = 0.1$ and time delay $\tau = 17$. This series predict the value $y(t+1)$ based on four previous measurements $y(t)$, $y(t-6)$, $y(t-12)$ and $y(t-18)$. The proposed neuron is trained with 450 samples and its prediction ability is tested

Table 4. Performance comparison with Mackey Glass time series data

S. No.	Network	Neuron Type	Parameters	MSE (Training)	MSE (Testing)	Epochs
1	4-3-1	Real-MLP	19	3.0×10^{-3}	3.8×10^{-3}	5000
2	4-1-1	$\mathbb{C}$-MLP	14	9.3×10^{-4}	1.02×10^{-3}	2000
3	4-1	$\mathbb{C}$-GPN	10	9.65×10^{-4}	1.23×10^{-3}	600

on other 500 samples. The performance of single proposed neuron for training and prediction data is presented in Fig.(2). Table 4 demonstrates that single $\mathbb{C}$-GPN outperform the two three layered network in terms of number of learning parameters and average training epochs.

4.2 Performance Evaluation with Three Layered Network

$\mathbb{C}$-XOR Problem. A set of training patterns using two rules in the experiment-2 of [11] has been described for performance evaluation of the 2-4-1 complex valued network comprising of summation neurons ($\mathbb{C}$-MLP). We pose here his definition as C-XOR problem. The comparative performance of different network structures for this $\mathbb{C}$-XOR problem is shown in Table 5 and in Fig.(3).

Functional Mapping. The general structure of mapping which is a composition of the three basic class of 2D transformations (*Scaling, Rotation and Translation*) is of the form :

$$\Psi(z) = Az + B \tag{15}$$

Where $z, A, B \in \mathbb{C}$. Evidently, this is an expansion or contraction by a factor $|A|$ and rotation through an angle equal to $ArgA$ in counterclockwise direction, followed by translation in a direction defined by the $ArgB$ through a distance equal to $|B|$. Learning and generalization of 2D transformations are only possible in complex valued neural network [11]. The three examples are presented in this section to validate the performance of generalized product neuron with complex back-propagation algorithm. We have taken 2-M-2 network where first input is a set of points lying on locus of a curve and second input is the reference point of input curve, similarly the first output neuron gives the locus of transformed curve and second output is its reference point. The hidden layer contains M proposed neurons. Transformations of the curves are shown in various figures in this section, *Black* color represents the input test figure, desired output is shown in dashed *Blue* color and actual output is shown in *Red* color.

Example 1. This example considers an experiment in which the points on a circle in z-plane get mapped to points on an image circle in w-plane. This mapping is used to study viscous flow across bodies with different cross-section. A 2-2-2 network is trained with 36 learning patterns. These 36 input data points are on

Table 5. Comparison of training and testing performance for $\mathbb{C}$-XOR-problem

Target error (MSE) = 0.0001

S.No.	Hidden Neurons	Average Epochs	Parameters	Misclassification
1	4-Sum	10000	34	0%
2	2-GPN, 2-RBF	9000	30	0%
3	4-GPN	6500	34	0%

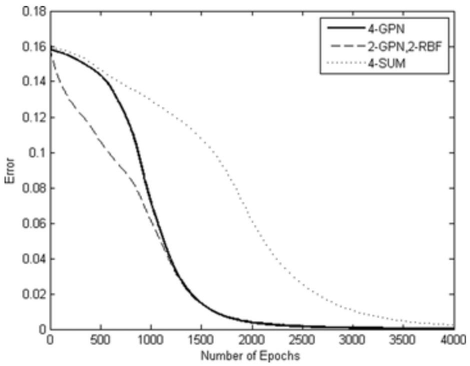


Fig. 3. Learning curves of three different 2-4-1 networks for ℂ-XOR

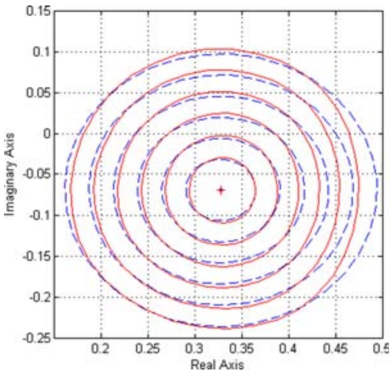


Fig. 4. Generalization performance of ℂ-GPN network for mapping in Eq. (16)

a circle $|z| = 0.62$ referenced at origin and output points are corresponding value of w defined by following equation :

$$w = \frac{(0.2z + 0.2 + 0.3i)}{(i + 0.4)} \tag{16}$$

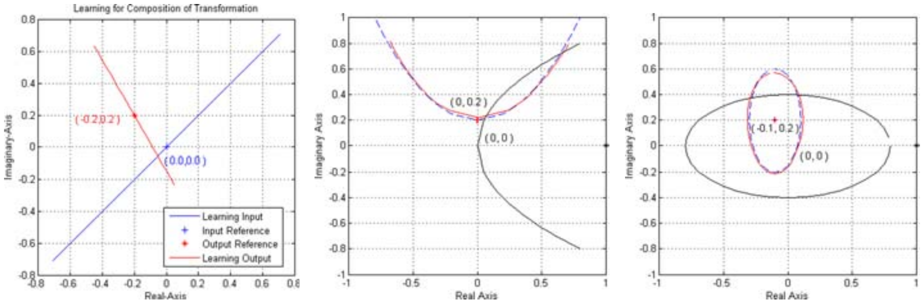
The trained network is able to generalize this mapping from z-plane to w-plane over varying value of radius of circles. Testing input patterns contain six circles of varying radius from 0.2 to 0.9 in regular interval and Fig.(4) presents the transformed result of these circles in w-plane.

In [11] author presented generalization ability of ℂ-MLP for each basic class of transformations but in our experiments we have considered the *composition of two or three basic transformations.* In examples (2) and (3), the 2-2-2 network is trained for input-output mapping, Fig.(5a), over a set of points lying on a line passing through a reference point (mid of line, may be origin) and generalization of this trained network is tested over other standard geometric curves like circle, ellipse, parabola; Fig.(5 b,c). The input training patterns are set of points of radius vector $r\,e^{i\vartheta}$. The corresponding output patterns are set of points $(\alpha\, r\, e^{i\,(\vartheta+\nu)}\; + \;\beta\, e^{i\,\sigma})$ representing the composition of three transformations, as in Eq.(15). All the points in training and testing patterns are with in a circle of unit radius centered at origin $(0 \leq r \leq 1)$ and all angles vary from 0 to $2\,\pi$. The author, in [11], discussed the generalization error and derived the equation for this error of transformation. The error increases as distance between the input test and training points increases and decreases as they become closer.

Example 2. Translation and Rotation
Here we investigate the behavior of the network which learned the composition of rotation and translation. From Eq.(15) :

$$\Psi 1(z) = Az + B \qquad where\,|A| = 1 \tag{17}$$

The mapping $\Psi 1$ rotates the vector z by $ArgA$ in counterclockwise direction and displaces by vector B.



(a) (b) (c)

Fig. 5. (a) Learning patterns for analysis, mapping shows scaling by factor α, angle of rotation τ and displacement by distance β in direction of σ (b) 2D Transformation (Translation and Rotation) with $\mathbb{C}$-GPN (c) 2D Transformation (Scaling, Translation and Rotation) with $\mathbb{C}$-GPN

The learning patterns z are rotated counterclockwise over $\pi/2$ radians and translated by $B = (0 + j \times 0.2)$. So such 19 training inputs lie on the line $y = x$, $(-0.636 \leq x \leq 0.636)$ and training output lie on the line $y = -x + 0.2$, $(-0.636 \leq x \leq 0.636)$. Fig.(5b) shows the generalization of trained network over parabola. Test input points lie on the parabola $y^2 = 4px$ with vertex at origin and focus at distance $p = 0.2$, test output should map on the parabola $x^2 = 4p(y - 0.2)$, which is due to rotation by $\pi/2$ and translation by (0,0.2).

Example 3. Scaling, Rotation and Translation
This example investigates the behavior of the network which learned the composition of all three transformations defined in Eq. (15). In this example the mapping Ψ performs scaling by factor $\alpha = 1/2$, rotation by $\pi/2$ radians counterclockwise and translation by $B = (-0.1 + j \times 0.2)$. The training set input have 21 patterns lying on line $y = x$, $(-1/\sqrt{2} \leq x \leq 1/\sqrt{2})$ referenced at origin and training output patterns lie on the line $y = -x + 0.1$, $(-0.453 \leq x \leq 0.253)$. The input test points lying on the ellipse $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$ would hopefully be mapped to points lying on $\frac{(x+0.1)^2}{(b/2)^2} + \frac{(y-0.2)^2}{(a/2)^2} = 1$ at reference (-0.1,0.2), where $a = 0.8, b = 0.4$. Fig.(5c) shows this generalization using $\mathbb{C}$-GPN neuron based network.

5 Conclusion

In this study we have developed a complex valued generalized product neuron ($\mathbb{C}$-GPN) based on polynomial aggregation function. The proposed neuron is simpler and reduces the complexity of learning parameters when used in complex valued neural network in compare to existing higher order neuron models. The proposed neuron model has been exploited as single neuron and as well as in the three layered network. All the simulations with $\mathbb{C}$-GPN demonstrate the significant improvement in the performance as compared to conventional neurons in terms of network structure, learning parameters and convergence speed. It is also observed that this single proposed neuron is able to solve many hard problems. This computational model may further be improved for better learning and approximation accuracy using efficient learning techniques.

References

1. Koch, C.: Biophysics of Computation: Information Processing in Single Neurons. Oxford University Press, Oxford (1999)
2. Giles, C.L., Maxwell, T.: Learning, invariance and generalization in a high-order neural network. Applied Optics 26(23), 4972–4978 (1987)
3. Koch, C., Poggio, T.: Multiplying with synapses and neurons, Single Neuron Computation, p. 315. Academic Press, Boston, Massachusetts (1992)
4. Li, D., Hirasawa, K., Hu, J., murata, J.: Multiplication units in feedforward neural networks and its training. In: Proceedings of ninth ICONIP 2002, vol. 1 (2002)
5. Piazza, F., Benvenuto, N.: On the complex Back propagation Algorithm. IEEE Transaction on Signal Processing 40(4), 967–969 (1992)
6. Box, G.E.P., Jenkins, G.M., Reinse, G.C.: Time series analysis: Forecasting and control. Prentice-Hall, Englewood Cliffs (1994)
7. Mackey, M., glass, L.: Oscillation and chaos in physiological control systems. Science 197, 287–289 (1997)
8. Schmitt, M.: On the Complexity of Computing and Learning with multiplicative Neural networks. Neural Computation 14, 241–301 (2001)
9. Durbin, R., rumelhart, D.E.: Product unit: A computationally powerful and biologically plausible extension to back-propagation networks. Neural Computation 1, 133–142 (1989)
10. Oh, S.K., Pedrycz, W., Park, B.J.: Polynomial neural networks architecture: analysis and design. In: Computers and Electrical Engineering, vol. 29, pp. 703–725. Elsevier Science, Amsterdam (2003)
11. Nitta, T.: An Extension of the Back-Propagation to Complex Numbers. Neural Networks 10(8), 1391–1415 (1997)
12. Nitta, T.: An analysis of the fundamental structure of complex-valued neurons. Neural Processing Letters 12, 239–246 (2000)
13. Chen, X., Li, S.: A Modified Error Function for the Complex-Value Back propagation Neural Network. Neural Information Processing 8(1) (2005)
14. Shin, Y., Ghosh, J.: The pi-sigma network: An efficient higher-order neural network for pattern classification and function approximation. In: Proceedings of the International Joint Conference on Neural Networks (1991)

Pulse-Type Hardware Neural Network with Two Time Windows in STDP

Katsutoshi Saeki[1,*], Ryo Shimizu[2], and Yoshifumi Sekine[1]

[1] College of Science and Technology, Nihon University
[2] Graduate School of Science and Technology, Nihon University,
7-24-1, Narashinodai, Funabashi-shi, Chiba, Japan
ksaeki@ecs.cst.nihon-u.ac.jp

Abstract. In recent years, synaptic plasticity, which is dependent on the order and time interval of pre- and post-synaptic spikes, has been observed by physiological experiments. There are two types of STDP which are characterized by an asymmetric time window and a symmetric time window (mexican hat type window). A symmetric time window especially depends on the influence of an inhibitory neuron. In this paper, we investigate the synaptic circuit and the synaptic weight control circuit using STDP by inhibitory interneuron input or no input. As a result, we show that the synaptic circuit using STDP with the time windows of these two types could be constructed with a simple circuit configuration considering an inhibitory interneuron by using the circuit simulator PSpice. Furthermore, we show the characteristic of reinforcement and suppression.

1 Introduction

In the brain, the learning is effected by the synaptic plasticity, which modulates and holds the connection weight. The brain has a superior information processing function when learning. For instance, the associative memory is one example. On the other hand, the present Neumann-type computer doesn't have a superior learning function like the brain. Then, an artificial neural network that performs similarly to the human brain would be required to construct a brain-type information processing system using analog VLSI technology [1].

Our purpose is to develop a brain-type information processing system using analog VLSI technology. Recently, the form of synaptic plasticity was seen to be dependent on the order and time intervals of pre- and postsynaptic spikes (STDP: spike timing dependent synaptic plasticity [2]) has been observed in the hippocampus and cerebral cortex [3, 4]. In general, STDP manifests itself as the potentiation of a synapse if the presynaptic spike precedes the postsynaptic spike and as depression if the presynaptic spike follows the postsynaptic spike (an asymmetric time window). In addition, it has been reported that the depression caused when the presynaptic spike precedes the postsynaptic spike (a symmetric time window) depends on the influence of an inhibitory neuron [5]. Namely, it has been reported that are two types of STDP which are characterized by an asymmetric time window and a symmetric time window (a

[*] Corresponding author.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 877–884, 2009.

mexican hat type window). Furthermore, it has been reported that recall of two states (autoassociative and heteroassociative) using the two time windows in the mathematical model [6] exist. We focus on the technological applications of the synaptic plasticity, we would like to develop on associative memory which makes use of the store and recall functions inherent in temporal sequences patterns, using the two time windows. Recently, it has been reported that a triplet rule, i.e. a rule which considers sets of three spikes [7] exists.

On the other hand, hardware neuron models with STDP have been proposed based on the results of physiological experiments [8, 9]. However, these models require either complex circuits or mixed signal circuits.

We previously proposed a simple cell body circuit of a pulse-type hardware neuron model (P-HNM) using CMOS that approximately simulates pulse signals as the means of information transmission in the brain [10, 11]. And so, we proposed an asymmetric type of STDP hardware model. Moreover, we studied the robustness of this circuit. As a result, we showed that it was able to make IC implementation [12, 13].

In this paper, we focus on the inhibitory inter neuron. We propose a pulse-type hardware neural network with two time windows in STDP by using circuit simulator PSpice.

2 Construction of Neural Network with STDP

Figure 1 shows a construction of a Neural Network with STDP. It shows a presynaptic cell N_{pre}, a post-synaptic cell N_{post}, and an inhibitory cell N_i between N_{pre} and N_{post}. Open circles indicate excitatory connections while solid circles indicate inhibitory connections. Also, stimulation currents i_{pre} and i_{post} flowing into N_{pre} and N_{post}, respectively, and $W_{post,\ pre}$, the weight of the synaptic connection between N_{pre} and N_{post}, are controlled by $V_{W_{post,\ pre}}$ from the synaptic weight generation circuit.

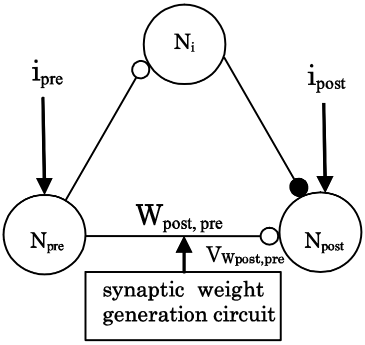


Fig. 1. Construction of a Neural Network with STDP

Figure 2 shows the circuit of a pulse-type hardware neural network with STDP. We used P-HNMs [10, 11] for N_{pre}, N_i and N_{post}. These P-HNMs have a threshold, an action potential, and a refractory period characteristic. And, N_{pre}, N_i and N_{post} are simple circuits because each circuit consists of four MOSFETs and two capacitors.

Furthermore, the substrate of M_{c1}, M_{c4}, and the gate terminal of M_{c2}, M_{c3} are connected to GND, because it receives the negative resistance characteristics [10, 11]. The circuit in Fig. 2 has two types of synaptic circuits, equivalent to the synaptic part in Fig. 1. These synaptic circuits have temporal summation characteristics. When voltage pulses v_{Npre} output from the pre-synaptic cell are input, the pulses have a first-order delay and are transmitted to M_{es1}, M_{es2} and M_{es3}, to generate i_{out} and i_{Ni}. Moreover, the current i_{out} can be adjusted by the voltage $V_{Wpost,\ pre}$, which is the output voltage of the synaptic-weight generation circuit. As well as the excitatory synaptic circuit, when the voltage pulses v_{Ni} output from the inhibitory synaptic cell are input, the pulses have a first-order delay and are transmitted to M_{is1}, M_{is2} and M_{is3}, to generate i_{in}. In other words, i_{in} suppresses the pulse (v_{out}) output by a P-HNM. In a synaptic weight generation circuit, D_{pre} is similar to D_{pre} in the excitatory synaptic circuit of Fig. 2, D_i is the same as D_i in the inhibitory synaptic circuit of Fig. 2, and D_{post} similarly shows the temporal summation of the input pulses. Whenever N_{pre} and N_{post} produce v_{Npre} and v_{Npost}, the voltage difference of capacitors C_{pre}, C_i, and C_{post} in D_{pre}, D_i, and D_{post} changes. They adjust the magnitude of the output currents from M_{pre2}, M_{i2}, and M_{post2}. Moreover, a multiple of the current in M_{pre2} flows through M_{pre3} and M_{pre6}, a multiple of the current in M_{i2} flows through M_{i3}, and a multiple of the current in M_{post2} flows through M_{post3}; in other words, they have current mirror structures. In addition, if v_{Npre} or v_{Npost} is inputted to the gate terminal, M_{pre7}, M_{i4}, and M_{post4} are switched ON. Therefore, $V_{Wpost,\ pre}$ is the output voltage of this circuit and is the parameter that controls the synaptic weight between the pre- and post-synaptic cells.

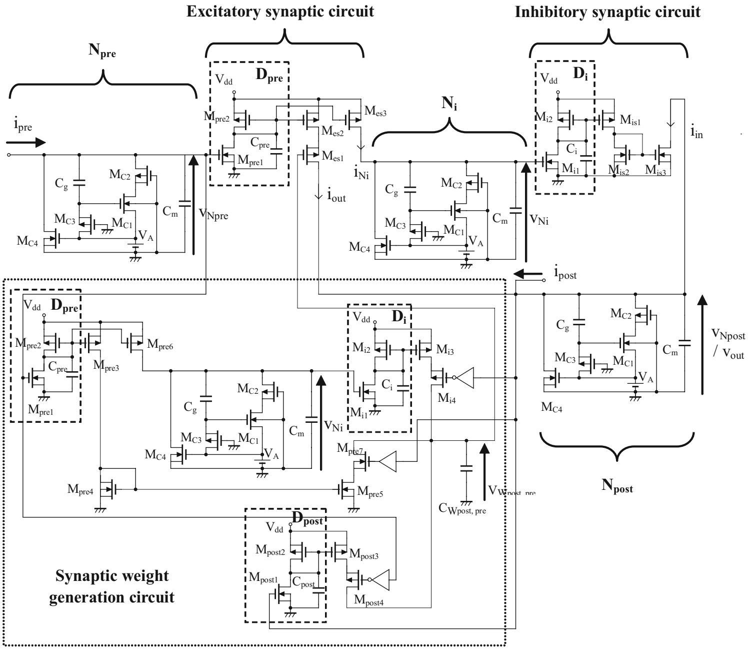
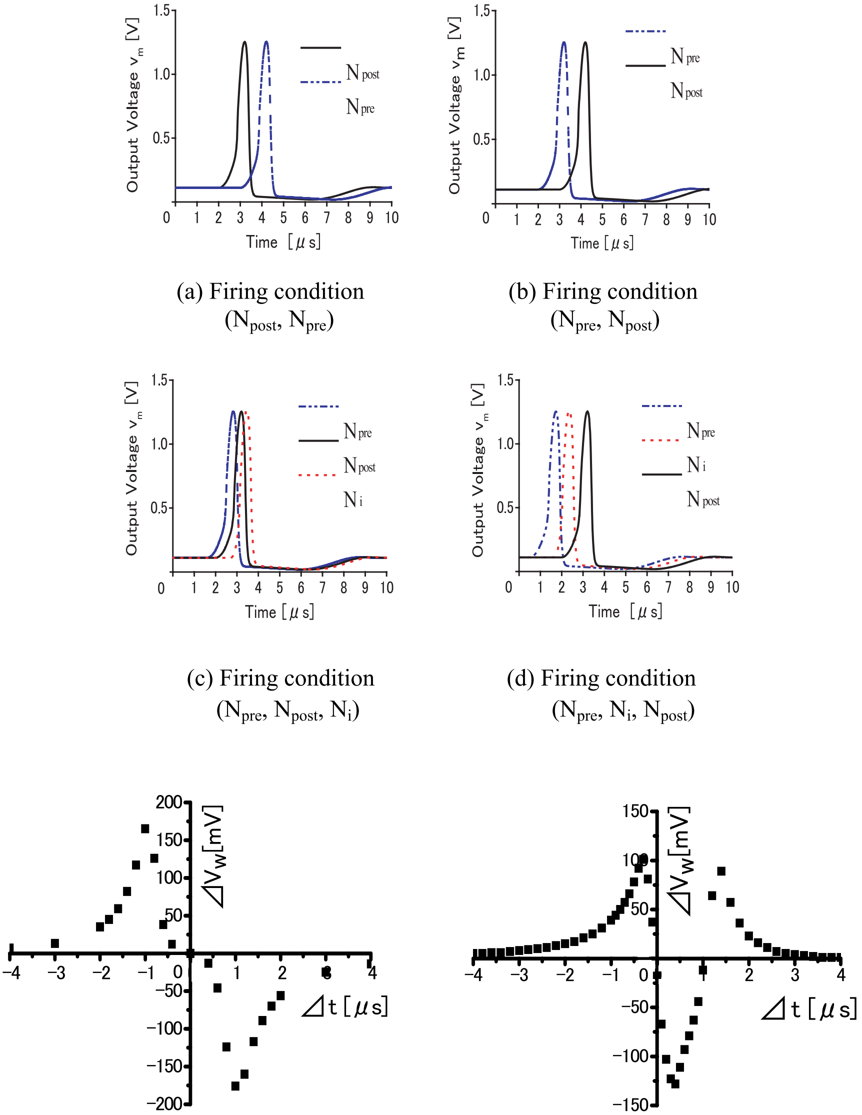


Fig. 2. Circuit of a pulse-type hardware neural network with STDP

3 Simulation Result

Figure 3 shows an example of two time windows in STDP. (a) ~ (d) show the timing of the pulse that the P-HNM used as each output voltage. Also, the output pulses of N_{pre} and N_{post} depend on the input timing of i_{pre} and i_{post}. The horizontal axis



(a) Firing condition (N_{post}, N_{pre})

(b) Firing condition (N_{pre}, N_{post})

(c) Firing condition (N_{pre}, N_{post}, N_i)

(d) Firing condition (N_{pre}, N_i, N_{post})

(e) Asymmetric time window

(f) Mexican-hat time window

Fig. 3. Example of two time windows in STDP

is time, and the vertical axis is the output voltage of each P-HNM. (e) and (f) show the behavior of $V_{Wpost,\ pre}$ in the synaptic-weight generation circuit. The horizontal axis is the time interval Δt, which is the time difference of the post-synaptic pulse and the pre-synaptic pulse, and the vertical axis is the amount of voltage change $\Delta V_{Wpost,\ pre}$. Here, $V_{Wpost,\ pre}$ increases after pulses are generated in the pre- and post-synaptic cells. The circuit parameters used were as follows: for M_{pre1}, M_{pre2}, M_{pre4}, M_{pre7}, M_{i1}, M_{i2}, M_{i4}, M_{post1}, M_{post2}, and M_{post4}, W/L=1.0; for M_{pre3}, M_{pre5}, and M_{post3}, W/L=1.1; for M_{pre6} and M_{c3}, W/L=0.1; for M_{i3}, W/L=0.8; for M_{c1} and M_{c2}, W/L=10; for M_{c4}, W/L=0.3; and C_{pre}=C_i=C_{post}=10 pF, $C_{Wpost,\ pre}$=15 pF, C_g=4 pF, C_m=1 pF, V_A=1.95 V, and V_{dd}=3 V. (a) and (b) show the suppression and reinforcement fields in Fig. (e), respectively. Furthermore, (a), (c) and (d) show the suppression (Δt=minus), reinforcement and suppression (Δt=plus) fields in Fig. (f), respectively. We showed that the circuit controls the synaptic weight, which is dependent on the order and the time interval of pre- and post-synaptic cell spikes, and we showed a similar characteristic to the asymmetric time window and the Mexican-hat time window found in the brain of a living body. Especially, the Mexican-hat time window is caused by an inhibitory interneuron lying between the pre- and post-synaptic cells.

Figure 4 shows $V_{Wpost,\ pre}$ vs. (a) output current i_{out}, (b) sink current i_{in} and (c) output voltage v_{out}, respectively. The horizontal axis is $V_{Wpost,\ pre}$, and the vertical axes are i_{out}, i_{in} and v_{out}, respectively.

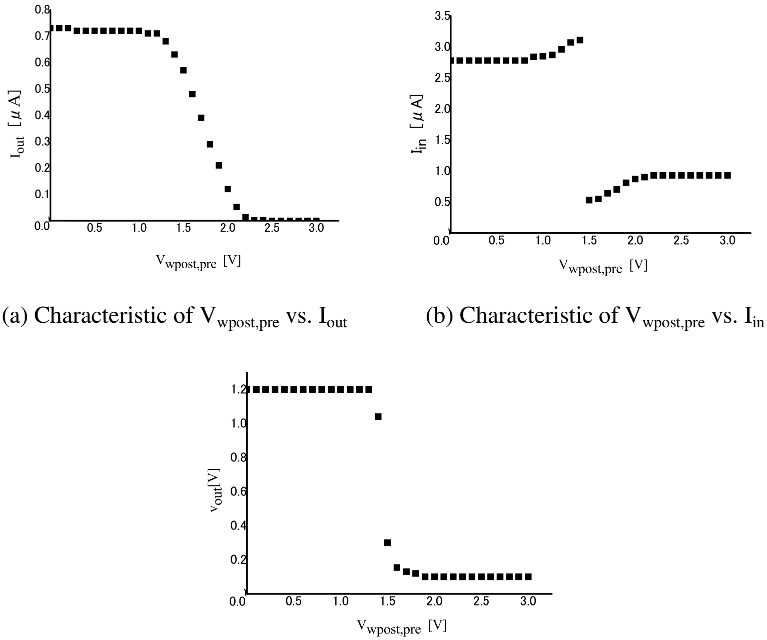
Figure 4(a) shows the amplitude characteristic of the output current i_{out} that flows from the excitatory synaptic circuit to the post-synaptic cell when the pre-synaptic cell generates the pulse to control the voltage of $V_{Wpost,\ pre}$. The circuit parameters used were as follows: for M_{pre1} and M_{pre2}, W/L=1.0; for M_{es1}, W/L=0.2; for M_{es2} and M_{es3}, W/L=0.1; C_{pre}=10 pF; and V_{dd}=3 V. It shows that i_{out} does not flow gradually as the control voltage of $V_{Wpost,\ pre}$ increases. At this point, i_{out} has a very low value (μA order), we can make a large scale neural network.

Figure 4(b) shows the amplitude characteristic of the sink current i_{in} that flows from the inhibitory synaptic circuit to the post-synaptic cell when the pre-synaptic cell generates the pulse to control the voltage of $V_{Wpost,\ pre}$. The circuit parameters used were as follows: for M_{i1} and M_{i2}, W/L=1.0; for M_{is1}, W/L=0.3; for M_{is2} and M_{is3}, W/L=0.5; C_i=10 pF; and V_{dd}=3 V. It shows that the amplitude of i_{in} decreases by about $V_{Wpost,\ pre}$, or 1.5 V, and i_{in} gradually becomes steady as the control voltage of $V_{Wpost,\ pre}$ increases.

Figure 4(c) shows the amplitude characteristic of the post-synaptic cell to control the voltage of $V_{Wpost,\ pre}$. It shows that the amplitude characteristic of the post-synaptic cell decreases gradually as the control voltage of $V_{Wpost,\ pre}$ increases.

As a result, we show two types of synaptic circuits, namely, an excitatory synaptic circuit and an inhibitory synaptic circuit, which can control i_{out} and i_{in} as the control voltage of $V_{Wpost,\ pre}$ changes. This shows that it is possible to control v_{out}, that is, the pulses output from the post-synaptic cell, with i_{out} and i_{in}.

Next, we discuss the characteristics of reinforcement and suppression of synaptic plasticity by using Fig.2. The signal of the next expression was used as a train of pulses that the cell body circuits N_{pre} and N_{post} output.



(a) Characteristic of $V_{wpost,pre}$ vs. I_{out}

(b) Characteristic of $V_{wpost,pre}$ vs. I_{in}

(c) Characteristic of $V_{wpost,pre}$ vs. v_{out}

Fig. 4. Characteristics of current and voltage in proposed circuit

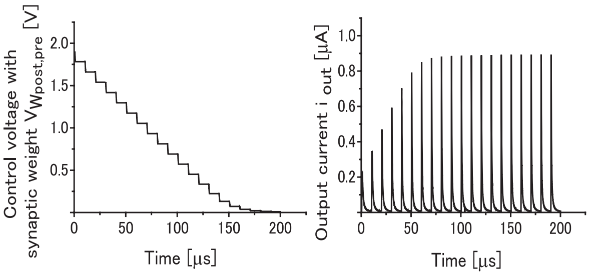
$$S_{pre} = \sum_{i=1}^{m} v_{pre,i}(T \cdot i) \tag{1}$$

$$S_{post} = \sum_{j=1}^{n} v_{post,j}(T \cdot j + dt) \tag{2}$$

In these equations, $v_{pre,i}$ represents the amplitude of the pulses of the i-th output from N_{pre}, and $v_{post,j}$, and shows the amplitude of the pulses of the j-th output from N_{post}. T is the period of the pulse, and dt indicates the delay time of the pulse train of N_{post} outputs to the pulse train of N_{pre} outputs. Based on the asymmetric time window, T=10 μs and dt=1 μs are used as a reinforcement pattern, while T=10 μs and dt=-1 μs are used as a suppression pattern.

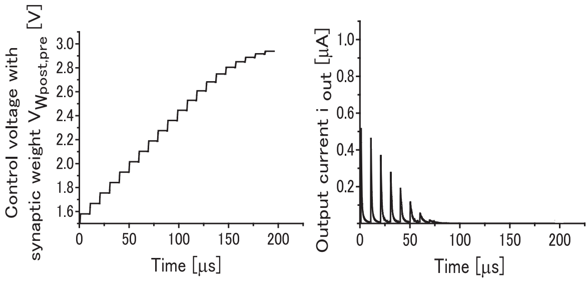
Here, S_{Npre} represents the periodic pulse train that N_{pre} outputs, and S_{Npost} shows the periodic pulse train that N_{post} outputs. First, $v_{Npre,\,i}$ is the amplitude of the i-th pulse that N_{pre} outputs, and $v_{Npost,\,j}$ is the amplitude of the j-th pulse that N_{post} outputs. T is the pulse period, and Δt is the delay time between the j-th pulse that N_{post} outputs and the i-th pulse that N_{pre} outputs. Examples of the time-varying characteristics in the weight increasing and decreasing process for Δt=0.5 μs (Fig. 5A), Δt=-0.5 μs (Fig. 5B), and Δt=1.5 μs (Fig. 5C) are shown in the Mexican-hat time window. We used a pulse period T=10 μs.

These results show that the synaptic weight of the excitatory synaptic circuit between the pre- and post-synaptic cells is increased or decreased by the control voltage of the synaptic-weight generation circuit.



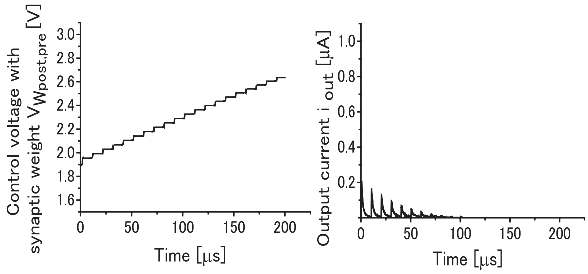
(a) $Vw_{post,pre}$ (b) i_{out}

Fig. 5A. Time transition in the increment process of Δt=0.5 [μs]



(a) $Vw_{post,pre}$ (b) i_{out}

Fig. 5B. Time transition in the inhibition process of Δt=-0.5 [μs]



(a) $Vw_{post,pre}$ (b) i_{out}

Fig. 5C. Time transition in the inhibition process of Δt=1.5 [μs]

4 Conclusion

In this paper, we investigated a pulse-type hardware neural network with two time windows in STDP. We showed a pulse-type hardware neural network can be constructed STDP with an asymmetric time window and a Mexican-hat time window.

Especially, we show that a pulse-type hardware neural network using STDP with a Mexican-hat time window can be constructed with a simple circuit configuration incorporating an inhibitory interneuron part.

In our future work, we will construct an integrated circuit with a pulse-type hardware neural network with STDP synapses. Furthermore, we will use the two time windows, we will construct a store and recall system of temporal sequences.

References

1. Mead, C.: Analog VLSI and Neural Systems. Addison-Wesley Pub. Co., Reading (1989)
2. Bi, G.-q., Poo, M.-m.: Synaptic modifications in cultured hippocampal neurons, Dependent on spike timing, synaptic strength, and postsynaptic Cell Type. J. Neurosci. 18, 10464–10472 (1998)
3. Roberts, P.D., Bell, C.C.: Spike Timingdependent synaptic plasticity in biological systems. Bio. Cybern. 87, 392–403 (2002)
4. Sakai, Y., Yoshizawa, S.: Mechanisms of synaptic competition and regulation in STDP rules. IEICE Technical Report, NC2003-21, pp. 1–6 (2003)
5. Tsukada, M., Aihara, T., Kobayashi, Y., Shimazaki, H.: Spatial analysis of spike-timing-dependent LTP and LTD in the CA1 area of hippocampal slices using optional imaging. Hippocampus 15, 104–109 (2005)
6. Samura, T., hattori, M., Ishizaki, S.: Autoassociative and heteroassociative hippocampal CA3 model based on location dependencies derived from anatomical and physiological findings. International Congress Series, vol. 1301, pp. 140–143. Elsevier, Amsterdam (2007)
7. Pfister, J.P.: Why are Triplets of Spike are Necessary in Models of STDP? In: Workshop on Plasticity at CNS (2006)
8. Bofill-I-Petit, A., Murray, A.F.: Synchrony detection and amplification by silicon neurons with STDP synapses. IEEE Trans. Neural Networks 15(5), 1296–1304 (2004)
9. Schemmel, J., Bruderle, D., Meier, K., Ostendorf, B.: Modeling Synaptic Plasticity within Networks of Highly Accelerated I&F Neurons. In: IEEE International Symposium on circuits and Systems, pp. 3367–3370 (2007)
10. Saeki, K., Sekine, Y., Aihara, K.: A Study on a Pulse-type Hardware Neuron Model using CMOS. In: Proc. International Symposium on Nonlinear Theory and Its Applications, vol. 2, pp. 855–858 (1999)
11. Sekine, Y., Sumiyama, M., Saeki, K., Aihara, K.: A Λ-Type Neuron Model using Enhancement-Mode MOSFETs. IEICE Trans (C) J84-C(10), 988–994 (2001)
12. Saeki, K., Hayashi, Y., Sekine, Y.: Extraction of Phase Information Buried in Fluctuation of a Pulse-type Hardware Neuron Model Using STDP. In: IEEE International Joint Conference on Neural Networks, pp. 2814–2819 (2006)
13. Saeki, K., Hayashi, Y., Sekine, Y.: Noise Tolerance of a Pulse-type Hardware Neural Network with STDP Synapses –Thermal Noise and Extraction of Phase Difference Information. In: Proc. IEEJ International Analog VLSI Workshop, pp. 88–93 (2007)

Time Evaluation for *WTA* Hopfield Type Circuits Affected by Cross-Coupling Capacitances

Ruxandra L. Costea and Corneliu A. Marinov

Department of Electrical Engineering
Polytechnic University of Bucharest, Romania
rux_co@itee.elth.pub.ro,cmarinov@rdslink.ro

Abstract. A continuous time neural network of Hopfield type is considered. It is a W(inner) T(akes) A(ll) selector. Its inputs are capacitively coupled to model the parasitics or faults of overcrowded chip layers. A certain parameter setting allows the correct selection of the maximum element from an input list. As processing time is a performance criterion, we infer upper bounds of it, explicitly depending on circuit and list parameters. Our method consists of converting the system of nonlinear differential equations describing the circuit to a system of decoupled linear inequalities. The results are checked by numerical simulation.

1 Introduction

The neural idea of properly connected simple cells performing complicated tasks, continues to attract and expand. In particular, continuous time Hopfield networks [1] [2] have proved their ability to process analog signals [3], [4], [5]. Having multiple stationary states which are stable and attainable by controllable dynamics, this type of circuits attracts equally circuit theorists and practitioners.

In this paper we consider a W(inner) T(ake) A(ll) circuit, i.e. a selector of a maximal element from a list. If the rank of the largest element of the input list is $\sigma(1)$, then the only output voltage above a positive threshold has the same rank $\sigma(1)$ [6]. In order to compete well with its digital counterparts, this circuit has to be able to signal fast "the winner" of a certain list then quickly reset and admit a new list. Setting up a proper clock is desirable and difficult as well. It requires an explicit time dependence on circuit and list parameters. To this goal, rate convergence estimations as they appear in stability studies, [7], [8] are not enough. Instead, recent developments deal with the individual component transient evolution [9], [10], [12], [13]. Our present paper extends these results by adding the capacitive coupling of the inputs to the original Hopfield model.

Motivated either by the parasitics inside the crowded VLSI chips or by possible faulty layers, the coupling capacitors strongly affect the processing speed [11]. We derive below analytical formulae of the switching time as functions of all circuit and list parameters and verify the results by numerical simulations [13].

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 885–892, 2009.

2 Preliminaries

We have N amplifiers with sigmoidal characteristics $v_i = Bg\left(\lambda u_i\right)$ where g is the *tanh* function. The gain λ is essential here. The cells are recurrently connected Fig. 1 - by the p value conductances gathered in the interconnection matrix T with $T_{ii} = 0$ and $T_{ij} = p$. C_0 collects the ground capacitances C_{int} of all interconnections arriving at each input: $C_0 = (N-1)\,C_{int}$. All pairs of inputs (u_i, u_j) are also connected by the cross-coupling capacitors δ. The matrix C with $C_{ii} = C_0 + (N-1)\,\delta$ and $C_{ij} = -\delta$ describes the capacitive part of the network.

If $u = (u_1, u_2, \cdots, u_N)^T$, $v = (v_1, v_2, \cdots, v_N)^T$, $b = (d_1 + M, d_2 + M, \cdots, d_N + M)^T$, $l = \dfrac{1}{\rho} + (N-1)\,p$ then the network is described by

$$C\frac{d}{dt}u = -lu - Tv + b \tag{1}$$

Very useful here is the ”difference equation”

$$C_n\frac{d}{dt}u_{ij} = -lu_{ij} + pv_{ij} + d_{ij} \tag{2}$$

where $C_n = C_0 + N\delta$, $u_{ij}\left(t\right) = u_i\left(t\right) - u_j\left(t\right)$ and similarly for v_{ij} and d_{ij}.

When S is in ”1” the list $d = (d_1, d_1, \cdots, d_N)^T$ and the bias current M are simultaneously applied to all N inputs. All through the T_p seconds - see Fig 2 - the network selects the maximal element of the list d. (The reasoning and the parameter restrictions leading to this WTA behavior are not detailed here - see [12] and [13].) This means that if $d_1 > d_2 > \cdots > d_N$, after T_p second we will have

$$v_1\left(T_p\right) > \xi > -\xi > v_2\left(T_p\right) > \cdots > v_N\left(T_p\right) \tag{3}$$

and

$$u_1\left(T_p\right) > \beta > -\beta > u_2\left(T_p\right) > \cdots > u_N\left(T_p\right) \tag{4}$$

where $\xi = Bg\left(\lambda\beta\right)$.

Once the result (3) or (4) has been read at T_p, the clock switches S in ”2” and keeps it there T_r seconds until the voltages u_i go beneath a threshold η. Below we only show how T_p is computed. However the η value is taken such that $T_r + t_{12}$

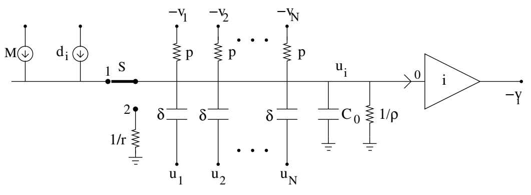


Fig. 1. The i-th cell with all its interconnections

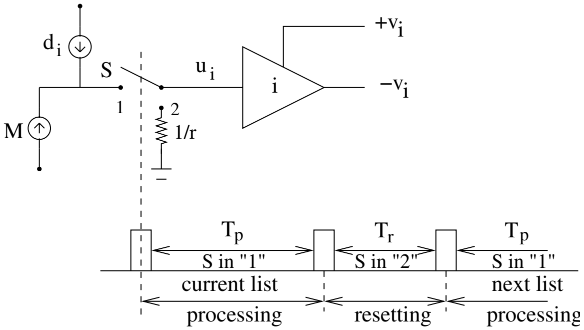


Fig. 2. The clock cycle

is the shortest possible - [10], [13]. Each list $(d_1, d_2, \cdots, d_N)$ is defined by its "separation" $\Delta = min|d_i - d_j|$, i.e. the shortest distance between two elements. If the admission interval is $[0, d_{max}]$ then we define the "relative separation" as $z = \dfrac{\Delta\,(N-1)}{d_{max}}$ a parameter between 0 and 1 describing "how crowded" are the list elements in $[0, d_{max}]$. Obviously the processing time depends on z. A value of T_p should be valid for any list having the relative separation higher than a limit z_{min}. Similarly the same clocking time T_p should work for any defect (or parasitic) capacitance smaller than a limit δ_{max}.

3 Computing the Processing Time

Once a list arrives at $t = 0$ the network voltages $u_i\,(t)$ and $v_i\,(t)$ start to move towards the new steady state $\overline{u}_i$ and $\overline{v}_i$. Those stationary values are ordered according to the d_i elements and the winner is above the value β while the losers lie under $-\beta$. In fact, the network reaches its WTA state at a certain moment t_p, following one of the two ways exposed in Fig. 3 and 4. If all other parameters are constant, t_p varies with:

- the defect (or parasitic) capacitance δ;
- the list separation Δ (or z);
- the size and succession of list elements at the same separation.

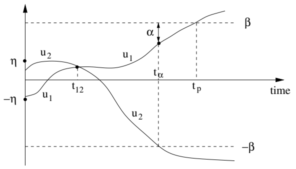


Fig. 3. The processing phase - case 1. The u_1 winner surpasses the threshold β after the moment when the losers $u_2 > u_3 > \cdots > u_N$ fall under $-\beta$.

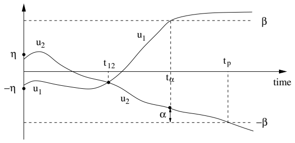


Fig. 4. The processing phase - case2. The u_1 winner goes above β before the moment when $-\beta = u_2 > u_3 > \cdots > u_N$.

In order to cover all these possibilities, the clock time T_p should be an upper bound of all times t_p with respect of all lists having $z \in [z_{min}, 1]$ and $\delta \in [0, \delta_{max}]$ where z_{min} and δ_{max} are given. As a matter of fact we split the "real" processing time t_p in three parts:

$$t_p = t_{12} + (t_\alpha - t_{12}) + (t_p - t_\alpha) \tag{5}$$

give an upper bound of each term (as explained below) and sum them up to obtain the desired T_p:

$$T_p = \bar{t}_{12} + \overline{t_\alpha - t_{12}} + \overline{t_p - t_\alpha} \tag{6}$$

Let us proceed explaining the three intermediate times in (5).

1. The first one begins at time zero when the new list arrives. There, $u_i\,(0)$ values are inherited from the previous list. The voltages evolve towards the new order. On their way, the moment t_{ij} when $u_i\,(t_{ij}) = u_j\,(t_{ij})$ after $u_i\,(0) < u_j\,(0)$ and before $u_i\,(t_p) > u_j\,(t_p)$, is an important one. By the method in [13] we can show that t_{ij} is unique. Also, from the difference equation (2) we can find an upper bound of t_{12}

$$\bar{t}_{12} = \frac{C_n^{max}}{\lambda p B - l} \ln \frac{1}{1 - \dfrac{\lambda p B - l}{\bar{l} - l}} \tag{7}$$

   Here $C_n^{max} = C_0 + N\delta_{max}$ and $\bar{l} = \dfrac{1}{\rho} + \dfrac{1}{r} + (N-1)\,p$.

2. If $\alpha \in [0, 2\beta]$ is arbitrary, let us denote by t_α the moment after t_{12} when one of the following happens: (see Figs 3 and 4)

$$u_1\,(t_\alpha) = \beta - \alpha \quad and \quad u_2\,(t_\alpha) = -\beta$$

   or

$$u_2\,(t_\alpha) = -\beta + \alpha \quad and \quad u_1\,(t_\alpha) = \beta$$

   The difference equation yields a bound of $t_\alpha - t_{12}$:

$$\overline{t_\alpha - t_{12}} = \frac{C_n^{max}}{l} \ln \frac{\Delta_{min}}{\Delta_{min} - l\,(2\beta - \alpha)} \tag{8}$$

   where Δ_{min} corresponds to z_{min}.

3. Referring to the first case of the t_α definition, we briefly explain below how we bound the $t_p - t_\alpha$ interval. The first equation in (1) can be written as

$$C_0 \frac{d}{dt} u_1 = -l u_1 - \delta \sum_{j=1}^{N} \frac{d}{dt} u_{1j} - p \sum_{j=2}^{N} v_j + d_1 + M \tag{9}$$

The terms $\frac{d}{dt} u_{1j}$ are evaluated from (2) written for $i = 1$. We get

$$\frac{d}{dt} u_{1j} \leq \frac{d_{1j} - l\,(2\beta - \alpha)}{C_n} \exp\left[-\frac{l}{C_n}\,(t - t_\alpha)\right] + \frac{2aB}{C_n} \tag{10}$$

On $[t_\alpha, t_p]$ interval we have

$$\begin{cases} -B < v_1 < \xi \\ -B < v_2, \cdots, v_N < -\xi \end{cases} \tag{11}$$

With (11) and (10), (9) gives the following linear differential inequality

$$C_0 \frac{d}{dt} u_1 \geq -l u_1 - A \exp\left[\left(-\frac{l}{C_n^{max}}\right) t\right] + R_1 \tag{12}$$

where $A = \frac{\delta_{max}}{C_n^{max}} \exp\left(\frac{l}{C_n^{max}} t_\alpha\right) \sum_{j=2}^{N} \left[d_{max} - (N - j)\,\Delta_{min} - l\,(2\beta - \alpha)\right]$, $R_1 = -\frac{\delta_{max}}{C_n^{max}} 2pB\,(N-1) + p\xi\,(N-1) + z_{min} d_{max} + M$.

After solving (12) we obtain

$$F_1\,(t_p - t_\alpha) \leq 0 \tag{13}$$

where

$F_1\,(s) = \left(\frac{R_1}{l} - \beta\right)\left[\exp\left(\frac{l}{C_0} s\right) - 1\right] - Q_1\,(\alpha)\left[\exp\left(\frac{l}{C_0} - \frac{l}{C_n^{max}}\right) s - 1\right] - \alpha$ and $Q_1\,(\alpha) = \frac{1}{lN} \sum_{j=2}^{N} \left[d_{max} - (N - j)\,\Delta_{min} - l\,(2\beta - \alpha)\right]$. An entirely parallel procedure for the second case of t_α definition leads to

$$F_2\,(t_p - t_\alpha) \leq 0 \tag{14}$$

F_2 has the form of F_1 where R_1 and Q_1 are replaced by $R_2 = -\frac{\delta_{max}}{C_n^{max}} 2pB\,(N-1) - p\,(N-2)\,B + p\xi - d_{max} + \Delta_{min} - M$ and $Q_2\,(\alpha) = \frac{1}{lN}\left\{d_{max} - (N-2)\,\Delta_{min} - l\,(2\beta - \alpha) - Z\,(\alpha) \sum_{j=3}^{N} \left[2pB + (j-2)\,\Delta_{min}\right]\right\}$ where

$$Z\,(\alpha) = \left[1 - \frac{l\,(2\beta - \alpha)}{\Delta_{min}}\right] \left[1 - \frac{\lambda pB - l}{\bar{l} - l}\right]^{\frac{l}{\lambda pB - l}}.$$

Let us suppose

$$\frac{R_{1,2}}{l} - \beta > 0 \tag{15}$$

Then, from (13) and (14) we obtain an upper bound of $t_p - t_\alpha$

$$\overline{t_p - t_\alpha} = max\left(s_1, s_2\right) \tag{16}$$

where s_1 and s_2 are roots of F_1 and F_2 i.e. $F_1\left(s_1\right) = 0$ and $F_2\left(s_2\right) = 0$. If we consider the parameters N, p, ρ, C_0, z_{min} and δ_{max} as given, and we suppose

$$\delta_{max} \leq \frac{C_0}{N-2} \tag{17}$$

then the restrictions (15) result in computable intervals for choosing M, β, λ, ξ and d_{max}. (Those formulae are omitted for lack of space - see [13]). Once these have settled, we have designed a circuit which correctly selects the maximal element and its computing time is related to parameters through (7), (8) and (16).

4 Examples

Let us consider our circuit with $N = 30$, $p = 0.1$, $\rho = 100$, $B = 10$, $C_0 = 1$, $z_{min} = 0.8$, $\delta_{max} = 0.017$. We compute $M_1 = -27.659$, $M_2 = -31.504$ and we choose $M \in [M_1, M_2]$ as $M = -31$. Further on $\beta_1 = 1.28 \times 10^{-2}$, $\beta_2 = 1.32 \times 10^{-2}$ and we choose $\beta \leq min\left(\beta_1, \beta_2\right)$ as $\beta = 1.28 \times 10^{-2}$. Then $\lambda = 638.06$, $\xi = 9.9999985$, $d_{max} = 3.39$. For each of the 11 values of $\alpha = (2\beta/10) \times n \in [0, 2\beta]$ we successively compute the bounds $\overline{t}_{12}$, $\overline{t_\alpha - t_{12}}$ from (7) and (8). Then we compute $Q_{1,2}\left(\alpha\right)$ for each of α values and $R_{1,2}$ as given in the preceding section. Then, for each α we solve the equations $F_{1,2}\left(s\right) = 0$ and find s_1 and s_2. By (16) we find $\overline{t_p - t_\alpha}$ and by (6) we obtain T_p for each α. The maximal T_p is the true one, $T_p = 3.34$. At this stage, all parameters (including the clock time T_p) have settled. We have built a circuit capable to select in $T_p = 3.34$ units of

Table 1. Successive steps in finding out T_p

n	0	1	2	3	4	5	6	7	8	9	10
$\alpha(\times 10^{-2})$	0	0.25	0.51	0.77	1.02	1.28	1.54	1.80	2.05	2.31	2.57
$\overline{t_{12}}$	8.33×10^{-5}										
$\overline{t_\alpha - t_{12}}$	0.94	0.71	0.56	0.44	0.35	0.27	0.20	0.14	0.09	0.04	0
R_1	5.29×10^{-2}										
$Q_1(\times 10^{-1})$	6.21	6.24	6.26	6.29	6.31	6.34	6.36	6.39	6.41	6.44	6.46
s_1	2.40	2.40	2.40	2.41	2.41	2.41	2.41	2.42	2.42	2.42	2.42
R_2	4.44×10^{-2}										
$Q_2(\times 10^{-1})$	-8.95	-9.05	-9.15	-9.26	-9.36	-9.46	-9.57	-9.67	-9.78	-9.88	-9.98
$s_2(\times 10^{-1})$	0	3.32	4.91	5.98	6.77	7.41	7.95	8.40	8.80	9.16	9.48
$max\left(s_1, s_2\right)$	2.40	2.40	2.40	2.41	2.41	2.41	2.41	2.42	2.42	2.42	2.42
T_p	**3.34**	3.12	2.97	2.85	2.76	2.68	2.62	2.56	2.51	2.46	2.42

Table 2. T_p for various $(z_{min}, \delta_{max}, N)$

$z_{min} \downarrow$	$\frac{\delta_t}{C_0} \rightarrow$	0.1	0.5	1
0.8	$N = 10$	3.29	8.34	8.46
	$N = 30$	1.93	3.34	1.21
	$N = 50$	1.23	1.82	1.30
0.5	$N = 10$	7.74	6.64	6.79
	$N = 30$	0.60	3.40	0.74
	$N = 50$	0.26	2.46	1.33
0.2	$N = 10$	6.36	8.43	7.58
	$N = 30$	1.10	0.91	1.25
	$N = 50$	0.35	1.71	1.39

Table 3. t_p from numerical simulation ODE solving

$z_{min} \downarrow$	$\frac{\delta_t}{C_0} \rightarrow$	0.1	0.5	1
0.8	$N = 10$	1.50	1.74	0.44
	$N = 30$	5.21×10^{-1}	4.29×10^{-1}	2.30×10^{-3}
	$N = 50$	2.94×10^{-1}	2.39×10^{-1}	9.16×10^{-4}
0.5	$N = 10$	1.01	0.49	0.098
	$N = 30$	4.40×10^{-2}	1.41×10^{-1}	1.35×10^{-3}
	$N = 50$	1.40×10^{-2}	7.85×10^{-2}	6.72×10^{-4}
0.2	$N = 10$	0.47	0.18	0.053
	$N = 30$	1.52×10^{-2}	1.60×10^{-3}	8.19×10^{-4}
	$N = 50$	4.80×10^{-3}	2.22×10^{-2}	3.26×10^{-4}

time the maximal element from any list of 30 elements with $z \geq 0.8$, even if the inputs suffer any $\delta \leq 0.017$ capacitive defect. Let us verify that our circuit works correctly. We take a list of 30 elements with $\varDelta = \varDelta_{min} = 9.35 \times 10^{-2}$ given by $d_i = (2i - 1)\,\varDelta$ for $i \in \overline{1, 15}$ and $d_i = (i - 16)\,2\varDelta$ for $i \in \overline{16, 30}$. Our circuit has a capacitive coupling of $\delta = \delta_{max}$. By running the ODE113 routine of MATLAB we found that u_{15} wins in $t_p = 0.42992$ units of time. This encodes the fact that d_{15} is the largest element of the list, a correct answer.

Tables 2 and 3 contain T_p and respectively t_p for circuits with different parameters, namely $N = 10, 30, 50$, $z_{min} = 0.2, 0.5, 0.8$ and $\delta_{max} = \dfrac{0.1}{29}, \dfrac{0.5}{29}, \dfrac{1}{29}$. (Note that $\delta_t = (N - 1)\,\delta_{max}$). All our examples confirm a correct WTA behavior in T_p units of time and $T_p > t_p$ everywhere. Sometimes in these examples the clock time T_p is much larger than the real time t_p. This does not mean a "waste of time" simply because different z and δ parameters can give a t_p closer to T_p. Indeed we are certain that T_p will exceed t_p regardless of $z \geq z_{min}$ and $\delta \leq \delta_{max}$.

5 Conclusion

This paper evaluates the processing time T_p for an analog WTA circuit by taking into account its capacitive coupling between input terminals.

The explicit formulae or simple equations relating different parts of T_p to circuit and list parameters are the main achievement here, hardly expected for such an intricate model. While t_{12} and $t_\alpha - t_{12}$ can be evaluated os in (7) and

([8]), $t_p - t_\alpha$ is bounded in ([16]) where s_1 and s_2 must be numerically computed from $F_1(s) = 0$ and $F_2(s) = 0$.

The result presented here suggests a procedure to set up the parameters (bias, threshold, gain, admission interval) enabling the circuit to correctly process lists of any density in T_p units of time regardless a capacitive defect $\delta \leq \delta_{max}$.

The results are limited by the supposition that the δ capacitance is the same between all pairs of inputs.

References

1. Hopfield, J.J.: Neurons with graded response have collective computational properties like those of two state neurons. Proc. Nat. Academy Sci. 81, 3088–3092 (1984)
2. Hopfield, J.J., Tank, D.W.: Neural computation of decisions optimization problems. Biol. Cybern. 52, 141–152 (1985)
3. Atkins, M.: Sorting by Hopfield nets. In: Proc. Int. Joint Conf. Neural Networks, Washington DC, vol. 2, pp. 65–68 (1992)
4. Dranger, T.S., Priemer, R.: Collective process circuit that sorts. IEE Proceedings-Circuits, Devices, Syst. 144, 145–148 (1997)
5. Cao, J.: Global exponential stability of Hopfield neural networks. Int. J. Syst. Sci. 32(2), 233–236 (2001)
6. Majani, E., Erlanson, R., Abu-Mostafa, Y.: On the K-winner-take-all network. In: Touretzky, D.S. (ed.) Advances in Neural Information Processing Systems, vol. 1, pp. 634–642. Morgan-Kafmann, San Mateo (1989)
7. Zhang, Y., Heng, P.A., Fu, W.C.: Estimate of exponential convergence rate and exponential stability for neural network. IEEE Trans. Neural Networks 10(6), 1487–1493 (1999)
8. Chen, T., Lu, W., Amari, S.: Global Convergence Rate of Recurrently Connected Neural Networks. Neural Computation 14, 2947–2957 (2002)
9. Calvert, B., Marinov, C.A.: Another K-winners-take-all analog neural network. IEEE Trans. Neural Networks 11, 829–838 (2000)
10. Marinov, C.A., Calvert, B.D.: Performance analysis for a K-Winners-Take-All analog neural network: Basic theory. IEEE Trans. Neural Networks 14, 766–780 (2003)
11. Cho, K.: Delay calculation capturing crosstalk effects due to coupling capacitors. Electronic Letters 41(8), 458–460 (2005)
12. Marinov, C.A., Hopfield, J.J.: Stable computational dynamics for a class of circuits with $O(N)$ interconnections capable of KWTA and rank extractions. IEEE Trans. Circuits and Systems, Part 1 52, 949–959 (2005)
13. Costea, R.L.: Artificial neural systems - Switching times for WTA circuits of Hopfield type. Doctoral Disertation, Polytechnic University of Bucharest (September 2007)

Circuit FPGA for Active Rules Selection in a Transition P System Region

Víctor Martínez, Abraham Gutiérrez, and Luis Fernando de Mingo

Natural Computing Group - Universidad Politécnica de Madrid – Madrid
Tel.: +34-91-336-7901; Fax: +34-91-336-7893
{victormh,abraham,lfmingo}@eui.upm.es

Abstract. P systems or Membrane Computing are a type of a distributed, massively parallel and non deterministic system based on biological membranes. These systems perform a computation through transition between two consecutive configurations. As it is well known in membrane computing, a configuration consists in a *m*-tuple of multisets present at any moment in the existing *m* regions of the system at that moment time. Transitions between two configurations are performed by using evolution rules which are in each region of the system in a non-deterministic maximally parallel manner. This article shows the development of a hardware circuit of selection of active rules in a membrane of a transition P-system. This development has been researched by using the Quartus II tool of Altera Semiconductors. In the first place, the initial specifications are defined in orfer to outline the synthesis of the circuit of active rules selection. Later on the design and synthesis of the circuit will be shown, as well as, the operation tests required to present the obtained results.

1 Theoretical Preliminaries on P-Systems

The Membrane Computing or P Systems (created by Păun [1], [2], [3]) are computation systems based on the biomolecular processes of living cells. According to this, the investigations are based on the idea of the imitation of the procedures that take place in Nature, and their application to machines, can lead to discover and to develop new computation models which will give place to a new generation of intelligent computers. There are many papers about software tools implementing different P-system variants [4], [5] and [6]. However, they are very interesting in order to define hardware implementation of these kinds of systems. Moreover, evolution of transition P- systems is very complicate to be translated into hardware devices due mainly to the membrane dissolving or membrane division capabilities of rules.

Besides that, the non-deterministic maximally parallel manner in which rules are applied inside membranes is more appropriated to be implemented in digital hardware devices. In the case of P-systems hardware implementations only a few papers can be found: a Hardware Circuit for Selecting Active Rules [7], a Cluster of Computers [8], a Master-Slave Distributed Architecture [9] or hardware architecture based on microcontrollers [10].

This article development a FPGA circuit for to select the active rules in a transition P-system membrane, by using the Quartus II tool of Altera Semiconductors [11]. In

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 893–900, 2009.

the first place, the initial specifications are defined to outline the circuit of the selection of active rules synthesis. Later on, the circuit synthesis will be shown as well as the functioning tests necessary to present the results we have reached.

The definition, according to formal notation, of the membrane system used for the implemented prototype is shown in the following expressions:

$$\Pi = (V,\mu,\omega_1,\ldots\omega_4,\ (R_1,\rho_1),\ldots,(R_4,\rho_4),4)$$

$$V = \{a,b,c,d,e\}$$

$$\mu = [_1[_2[_3]_3]_2[_4]_4]_1$$

$$\omega_1 = aac$$

$$R_1 = \{r_1\text{: } c \rightarrow (c,in_4),\ r_2\text{: } c \rightarrow (b,in_4),\ r_3\text{: } a \rightarrow (a,in_2)b,\ r_4\text{: } dd \rightarrow (a,in_4)\}$$

$$\rho_1 = \{\ r_1 > r_3,\ \ r_2 > r_3\}$$

$$\omega_2 = a$$

$$R_2 = \{r_1\text{: } a \rightarrow (a,in_3),\ r_2\text{: } ac \rightarrow \delta\}$$

$$\rho_2 = 0$$

$$\omega_3 = cd$$

$$R_3 = \{r_1\text{: } a \rightarrow \delta\}$$

$$\rho_3 = 0$$

$$\omega_4 = \lambda$$

$$R_4 = \{r_1\text{: } c \rightarrow (d,out),\ r_2\text{: } b \rightarrow b\}$$

$$\rho_4 = 0$$

The circuit to be carried out obtains the region 1 active rules. The region 1 input values will be the objects multiset w_1, the group of rules R_1, the priority relationships among rules ρ_1 and the inner adjacent regions number (two regions in this case). The objects number V is 5 with decimal multiplicity (0 - 9), therefore, we will need 4 bits to represents each object. And so, a word representing the multiplicity of the 5 objects will occupy 20 bits. The rules group is formed by 4 rules which will be stored in the device memory.

Each of the 4 rules we need to store in the memory will be coded with 64 bits. This is, 20 bits for the antecedent, 20 bits for each consequent of the two inner interior regions, plus 4 remaining bits used to code the priorities mask of each rule with regard to the other ones. The rules are stored, therefore, in a ROM 4x64 bits memory.

2 Hardware Implementation of Active Rules

The logical circuit general description is obtained by means of the conjunction of certain fundamental elements. These elements have been denominated basic Functional Units (*FU's*). These FU's controls the positive validation of a certain evolution rule in function of its applicability, utility and activation properties. The Fig. 1 show the general outline of the circuit based on these functional units.

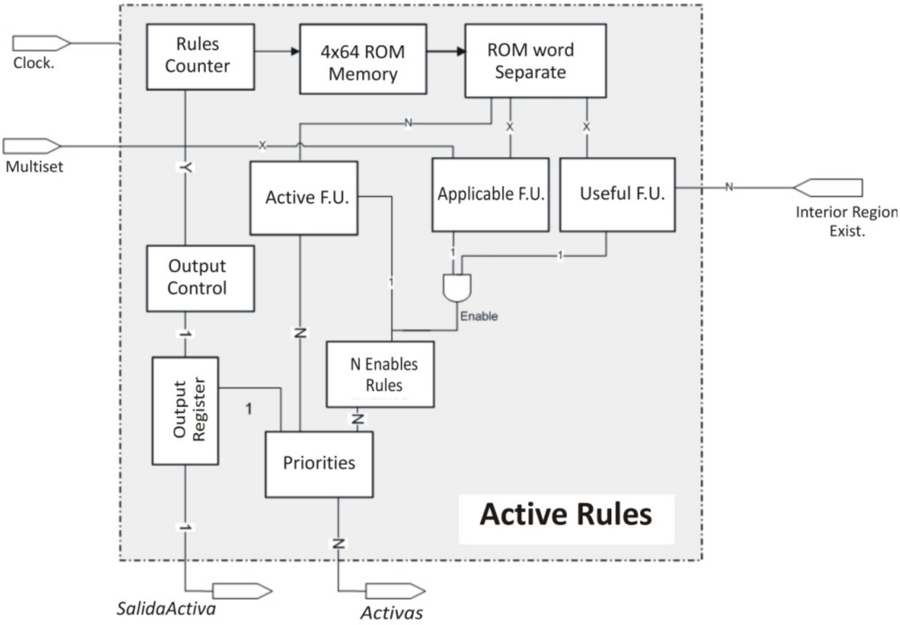


Fig. 1. Scheme general of the circuit to obtain the active rules of a membrane region

Basically, the global circuit inputs are composed by the region objects multiset of the membrane to be evaluated, the in_i bits to determine the interior regions existence, an input clock that allows going on reading and processing the memory rules and a *set* input to initialize the active rules register of the circuit. The register output shows, in positive logic, the active rules of the region, after processing all the ROM memory rules.

In the circuit operation, the memory words will be sequentially read and the different parts that compose each word will be separate. Each different part will be used in the corresponding functional unit or sub-circuit. A counter circuit is needed to read sequentially the memory words. A distributor circuit will be used to separate the different groups of each word.

The usable rule condition is checked by the *Useful Functional Unit* circuit. According to this condition, this particular circuit should check, in each rule, whether the consequent *v* of the rule has some objects which are sent to the inner adjacent regions. In this case, the interior region should exist. The applicability condition of an evolution rule is performed by the *Applicable Functional Unit* circuit. According to the which, the objects of its antecedent should exist in the region multiset in the same or bigger multiplicity.

The *Active Functional Unit* is the part of the circuit in charge of evaluating the priorities among all the existing rules in the membrane region. The result obtained in this unit is a N bits output vector (being N the number of rules). This vector contains the rules potentially active (bit=1) and the rules that should be inhibited in the case of being useful and applicable a rule of more priority that disables it (bit=0).

3 Experimental Results

Next we will present the results of simulating the selection of active rules circuit of the P system example presented in the section 1. For this example, the circuit will obtain the active rules of the region 1, being its two interior regions the 2 and the 4 ones.

On the other hand, the software Quartus II of Inc Altera includes a simulator that can be used to verify as much the behavior as the yield of the carried out designs. The Quartus II simulator allows defining input vectors that will serve as stimulus to the circuit inputs. The application of the input vectors in the simulation allows checking the state of the exits of the circuit, verifying the validity, functionality and effectiveness of the circuit this way.

In Quartus II, we can perform functional simulations, where the circuit output exclusively depends on the inputs values, or we can carry out simulations of a major complexity and depending on time, where one or more clock signals are defined. In our particular case, we are carried out the sequentially reading of the rules defined in ROM memory, with the help of a system clock implemented for that purpose.

To be able to simulate the circuit, the previously described steps for the *Design and Synthesis of the Circuit of Active Rules* must be fulfilled completely:

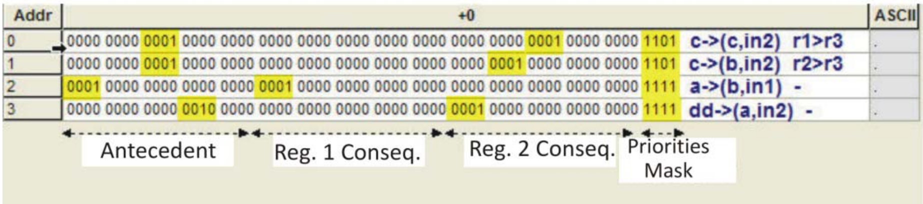
- Creation of the project "*ActiveRules*"
- Inclusion in the project of all the files with VHDL descriptions of the different elements of the design.
- Inclusion in the project of the schematic blocks files with the design of the circuit.
- Election of the device destination: Cyclone II EP2C35F672C6.
- Circuit Compilation (Synthesis).

As a previous step to the simulation execution, it is necessary to load in the ROM memory system the code of the defined rules for the P system region that is going to be simulated. In this case, the information relative to the rules group R_1, as well as the priorities ρ_1 will be recorded in the ROM memory circuit. The consequents of the regions *here* and *out*, are not necessary for the functionality of the circuit, since they are not evaluated to determine whether a rule is applicable or not.

The information needed to code the region rules, for a certain state, will loaded in words of 64 bits, according to the following distribution:

1. The 20 bits most significant (bits 63 to 44) contain the rule antecedent.
2. The following 20 bits (bits 43 to 24) contain the consequent for the inner region 1.
3. The next 20 bits (bits 23 to 4) contain the consequent for the inner region 2.
4. The last 4 bits of each word of the ROM are used to establish the mask of priorities among the rules defined in the membrane.

In the circuit operation phase, once all the rules have been read sequentially and its evaluation (several clock pulses are needed to address the memory ROM words) has been completed, the *Active Functional Unit* circuit will be responsible for the composing of all the priorities among the rules defined in the system, and for the the



Addr	+0	ASCII
0	0000 0000 0001 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 0000 0000 1101 c->(c,in2) r1>r3	.
1	0000 0000 0001 0000 0000 0000 0000 0000 0000 0000 0000 0001 0000 0000 0000 1101 c->(b,in2) r2>r3	.
2	0001 0000 0000 0000 0000 0001 0000 0000 0000 0000 0000 0000 0000 0000 0000 1111 a->(b,in1) -	.
3	0000 0000 0000 0010 0000 0000 0000 0000 0000 0000 0001 0000 0000 0000 0000 1111 dd->(a,in2) -	.

Fig. 2. Shows the method adopted to stored information into the 4x64 ROM memory of the rules R_1 and the priorities masks of the region *1* in the membrane example

building up of a final mask of priorities, which will be applied to the output vectors proceeding from the other two functional units.

Finally, to check the behavior of the designed circuit, the simulation (*'waveforms'*) vectors are created. They represent the type, the forms and the characteristics of the input signals. In the same way, in the simulation vector the signals representing the output of the circuit we want to evaluate are identified. The simulator applies the test vectors to the compiled design and determines the resulting values in the system outputs.

The next step is to include the circuit inputs and outputs going to be simulated:

- '*Set'*: (input, 1 bit). Initialization input of circuit components (ROM, rules counter, Functional Units....)
- *'In1'*: (input, 1 bit). Existence of internal region 1.
- '*In2'*: (input, 1 bit). Existence of internal region 2.
- *"w_a", "w_b", "w_c", "w_d", "w_e"*: (input, unsigned decimal, 4 bits). Input Multiset elements.
- '*Clk'*: (input, 1 bit). System clock.
- '*Activas'*: (output, 4 bits). Exit of the system representing the active rules.
- *'Salida Activa'* (output, 1 bit). Flag determining when the active rules output is stable.

The test case shown corresponds with a computation model in which the circuit of active rules works together with other elements that were changing their conditions according to the temporary evolution of the system. So, we can introduce new aleatory values in the input vectors, by using the *'Set'* flag.

We can check how the *'SalidaActiva'* is validated in irregular periods (due to the *'Set'* activations), and how it becomes stable when there are not new loads of data. The chronogram appeared in figure 3 shows the variation of the inputs signals and the outputs result.

The initial values are:

- w_a [3..0] = 2 (decimal) --> Applicable r_3
- w_c [3..0] = 1 (decimal) --> Applicable r_1 and r_2
- w_d [3..0] = 3 (decimal) > Applicable r_4
- w_b, w_e = 0
- In1 = 1 (Region 1 Exists) --> r_3 is Useful

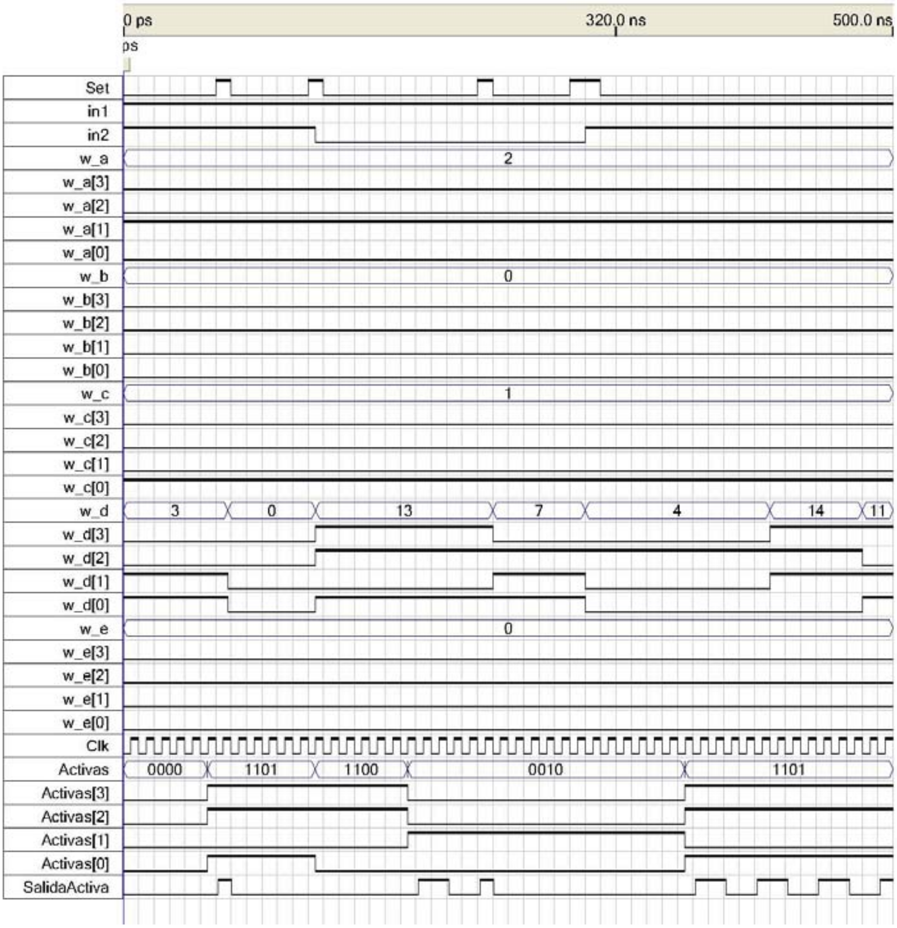


Fig. 3. Inputs-output signals variation

- In2 = 1 (Region 2 Exists) --> r_1, r_2 and r_4 are Useful
- Clock --> Cycle 10ns

Therefore, in accordance with these conditions, the rules r_1, r_2, r_3 and r_4 are all useful and applicable initially. The priorities mask will be 1101, this mask disables r_3. The initial active rules output will have the 1101 value after 5 clock periods showing that the r_1, r_2, and r_4 rules are actives.

An external initialization (*'Set'*) changing the input vectors values to *w_d [3..0] = 0*, would make r_4 not applicable. In this case, the signal *'SalidaActiva'* is annulled by a new change of the *'Set'* signal and the load of new input values. It can be proved how, once the 50ns necessary to complete a new calculation have passed, the change takes place in the vector *'Activas'* and one period later it becomes valid with a 1 in the *'SalidaActiva'* signal. The obtained values remain stable until a new activation of *'Set'* is produced.

4 Conclusions and Future Remarks

This paper presents a direct way to obtain a FPGA circuit capable to select the active rules inside the P-system membrane. The final step is to implement a hardware circuit accomplishing the outlined initial requirement. That is, given an initial multiset of objects, a finite set of evolution rules and an initial Active Rules, the circuit provides the set of rules to be applied to a membrane. The development of the digital system can be carried out by using hardware-software architectures as schematic blocks or VHDL language. The physical implementation can be accomplished on hardware programmable FPGA's devices.

When analyzing the objectives reached, it can be proved that both the designs as much as accomplished synthesis, allow to obtain a hardware system for a general model of P transition system. This means that we can reach the behavior of a region of any membranes system with the obtained circuit. The only limitation is the maximum size of certain parameters, such as the number of rules, the multiplicity of the objects or the number of interior adjacent regions.

The obtained circuit behaves based on the evolution rules stored in memory (which do not change during the system evolution process) and the inputs, which correspond with the values of the region state (reflected in its objects multiset).

If the conditions of the region change, the circuit modifies its outputs being adjusted to the new values. This feature is of a supreme importance in order to integrate this circuit as a module that works cooperatively together with other circuits. These circuits can carry out the rest of the tasks needed to complete the evolution of a transition P system.

References

1. Păun, G.: Computing with Membranes. Journal of Computer and System Sciences 61, 108–143 (2000); Turku Center of Computer Science-TUCS Report No 208 (1998)
2. Păun, G.: Computing with membranes. An introduction. Bulletin of the EATCS 67, 139–152 (1999)
3. Păun, G.: Membrane Computing. Basic Ideas, Results, Applications. In: Pre-Proceedings of First International Workshop on Theory and Application of P Systems, Timisoara, Romania, September 26-27, 2005, pp. 1–8 (2005)
4. Arroyo, F., Luengo, C., Baranda, A.V., Mingo, L.F.: A software simulation of transition P systems in Haskell. In: Pre-Proceedings of Second Workshop on Membrane Computing, Curtea de Arges, Romania, August 2002, pp. 29–44 (2002); Păun, G., Rozenberg, G., Salomaa, A., Zandron, C. (eds.) WMC 2002. LNCS, vol. 2597, pp. 19–32. Springer, Heidelberg (2003)
5. Arroyo, F., Luengo, C., Fernandez, L., Mingo, L.F., Castellanos, J.: Simulating membrane systems in digital computers. International Journal Information Theories and Applications 11(1), 29–34 (2004)
6. Fernández, L., Arroyo, F., Castellanos, J., Martinez, V.J.: Software Tools / P Systems Simulators Interoperability. In: Pre-proceedings of the 6th Workshop on Membrane Computing, Vienna, Austria (July 2005)

7. Víctor, J., Martínez, L., Fernández, F., Arroyo, A., Gutiérrez: A Hw Circuit for the Application of Active Rules in a Transition P-System Region. In: Fourth International Conference Information Research and Applications I.TECH 2006, Varna, Bulgaria, Del 20 al 25 de Junio de (2006)
8. Ciobanu, G., Guo, W.: P Systems Running on a Cluster of Computers. In: Martín-Vide, C., Mauri, G., Păun, G., Rozenberg, G., Salomaa, A. (eds.) WMC 2003. LNCS, vol. 2933, pp. 123–139. Springer, Heidelberg (2004)
9. Bravo, G., Fernández, L., Arroyo, F., Tejedor, J.A.: Master-Slave Distributed Architecture for Membrane Systems Implementation. In: 8th wseas International Conference on Evolutionary Computing, EC 2007 (2007)
10. Gutiérrez, L., Fernández, F., Arroyo, V., Martínez: Design of a hardware architecture based on microcontrollers for the implementation of membrane system. In: Proceedings on 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2006), Timisoara, Rumania, September 2006, pp. 39–42 (2006)
11. Altera Corporation, Silicon Valley, `http://www.altera.com/`

Part VII

Machine Learning and Information Algebra

Model Selection Method for AdaBoost Using Formal Information Criteria

Daisuke Kaji[1,2] and Sumio Watanabe[3]

[1] Computational Intelligence and System Science, Tokyo Institute of Technology,
Mailbox R2-5, 4259 Nagatsuda, Midori-ku, Yokohama 226-8503, Japan
[2] Konicaminolta Medical & Graphic, INC. 2970 Ishikawa-machi, Hachioji-shi, Tokyo
192-8505, Japan
[3] Precision and Intelligence Laboratory, Tokyo Institute of Technology, 4529
Nagatsuda, Midori-ku, Yokohama 226-8503, Japan

Abstract. AdaBoost is being used widely in information systems, artificial intelligence and bioinformatics, because it provides an efficient function approximation method. However, AdaBoost does not employ either the maximum likelihood method or Bayes estimation, and hence its generalized performance is not yet known. Therefore an optimization method for the minimum generalization error has not yet been established. In this paper, we propose a new method to select an optimal model using formal information criteria, AIC and BIC. Although neither AIC nor BIC theoretically corresponds to the generalization error in AdaBoost, we show experimentally that an optimal model can be chosen by formal AIC and BIC.

1 Introduction

AdaBoost is a learning algorithm enabling the construction of an optimal ensemble of weak classifiers for minimum training errors. Since it performs well in function approximation, it is being used in pattern recognition, artificial intelligence, and bioinformatics[2][4]. AdaBoost reduces the training error; however this fact does not necessarily imply that the generalization error is also small. The generalization performance of AdaBoostis still unclear, because it is the different algorithm from the maximum likelihood method and Bayes estimation.

In this paper, we propose that formal AIC and BIC can be used as information criteria for AdaBoost model selection, based on the idea that the risk function used in AdaBoost is a kind of probabilistic learning machine. The formal information criteria AIC and BIC are defined by the assumption that the optimized ensemble constructed using AdaBoost approximately minimizes the risk function. Although such an assumption is not satisfied in general, we show experimentally that the proposed method yields an appropriate model with a small generalization error. Using the formal AIC, the average generalization error is reduced, whereas with the formal BIC, the worst generalization error is reduced.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 903–910, 2009.

2 AdaBoost Algorithm

AdaBoost is one of the forward stagewise modeling, which adds a new decision (classifier) at each stage to the latest results so as to obtain an optimal decision. The algorithm is derived from successive minimization of the risk function using the coordinate descent method[4]. Although AdaBoost is generally presented as a two-class classifier, we describe here AdaBoostMlt, which is an algorithm for a version of AdaBoost with multi-class classification[2] and we also use this algorithm for the experiments that we examine later.

Firstly, we denote $\mathcal{W}^{mlt} = \{h^{mlt} : X \to 2^Y\}$ as the set of maps from the input data set $\boldsymbol{x} \in X \subset \mathbb{R}^n$ to the power set 2^Y of the output class $Y = \{1, \cdots, K\}$; then we define the set of weak classifiers

$$\mathcal{W} = \{h : X \times Y \to \mathbb{R} \mid h(\boldsymbol{x}, y) = [y \in h^{mlt}(\boldsymbol{x})], h^{mlt} \in \mathcal{W}^{mlt}\},$$

where $[\cdot]$ is a function which returns 1 if "$\cdot$" is true, and returns 0 in other cases. We also consider the following empirical distribution of training data $(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_N, y_N)$, defined by

$$p_0(\boldsymbol{x}, y) = \begin{cases} \frac{1}{N} & (\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_N, y_N) \\ 0 & (\text{other}) \end{cases} .$$

and we define

$$p_0(\boldsymbol{x}, y, y') = \frac{1}{K-1} p_0(\boldsymbol{x}, y)$$

as the extended joint probability which is extended formally to the error class y'. In addition, we define the classification error function $e(h, p)$ of a weak classifier h, the total score of correct and incorrect classifications $p_+(h, p), p_-(h, p)$ and the risk function L as follows:

$$e(h, p) = \sum_{\boldsymbol{x}, y, y'} p(\boldsymbol{x}, y, y')(I(h(\boldsymbol{x}, y) - h(\boldsymbol{x}, y'))),$$

$$p_+(h, p) = \sum_{\boldsymbol{x}, y, y'} p(\boldsymbol{x}, y, y')[h(\boldsymbol{x}, y) - h(\boldsymbol{x}, y') > 0],$$

$$p_-(h, p) = \sum_{\boldsymbol{x}, y, y'} p(\boldsymbol{x}, y, y')[h(\boldsymbol{x}, y) - h(\boldsymbol{x}, y') < 0],$$

$$L(f) = \frac{1}{m(K-1)} \sum_{1 \le i \le m} \sum_{y' \ne y_i} E(\boldsymbol{x}_i, y_i, y'),$$

where we are using

$$I(z) = \begin{cases} 1 & (z > 0) \\ 0 & (z = 0) \\ -1 & (z < 0) \end{cases}$$

and

$$E(\boldsymbol{x}_i, y_i, y') = \exp\{-(f(\boldsymbol{x}_i, y_i) - f(\boldsymbol{x}_i, y')\} .$$

Finally, we denote the classifier resulting from a linear combination of weak classifiers $h_1, \cdots, h_T$ (T:learning times) thus:

$$f_{\boldsymbol{\lambda}}(\boldsymbol{x}, y) = \lambda_1 h_1(\boldsymbol{x}, y) + \cdots + \lambda_T h_T(\boldsymbol{x}, y) .$$

Then the algorithm of AdaBoost is given by Fig.1. It is clear from the algorithm that AdaBoost is applicable to any set of weak classifiers. This means that to apply the algorithm requires us to construct a set of weak classifiers in advance. To assist us, we can utilize the decision stumps, which is a useful method for constructing weak classifiers. The decision stumps are weak classifiers which decide the classification result by the value along one dimension of the training data vector. We provide an explanation of the decision stumps in the next section.

```
1. Input: n samples ${(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_n, y_n)}$;
2. Initialize : $\boldsymbol{\lambda}_0 = (0, \cdots, 0)$. Set $p_0$ as the extended joint probability;
3. Do the process below for $t = 1, \cdots, T$:
   1) Select the weak classifier $h_t$ with minimum error $e(h_t, p_{t-1})$;
   2) $\boldsymbol{\lambda}_t \leftarrow \boldsymbol{\lambda}_{t-1} + (0, \cdots, \alpha_t, \cdots, 0)$ for $\alpha_t = \frac{1}{2}\log\frac{p_+(h_t; p_{t-1})}{p_-(h_t; p_{t-1})}$;
   3) If $L(f_{\boldsymbol{\lambda}})$ does not chenge then break;
   4) Update the extended joint probability $p_t$:
         $p_t(\boldsymbol{x}_i, y_i, y') \leftarrow \frac{1}{Z}\exp\frac{f_{\boldsymbol{\lambda}_t}(\boldsymbol{x}, y') - f_{\boldsymbol{\lambda}_t}(\boldsymbol{x}, y_i)}{m}$   ($Z$ : normalizing constant);
   5) $t \leftarrow t + 1$;
4. Output: $p(y|\boldsymbol{x}) = \frac{1}{Z}e^{f_{\boldsymbol{\lambda}_t}(\boldsymbol{x}, y)}$;
```

Fig. 1. AdaBoostMlt algorithm

3 Weak Classifiers Using the Decision Stump

The decision stumps are one of the simplest weak classifiers which make a decision based on one element x_i of the input vector $\boldsymbol{x} = (x_1, \cdots, x_n)$. For instance, if the value of x_i is more than a certain threshold value, then the classifier judges this data to be a member of class k. Therefore the decision stumps are denoted by $h^+_{i,th,k}, h^-_{i,th,k}$, where i is coordinate of focus, th is threshold value, k is an index of class and ”+” means when $x_i \geq th$ then $x \in$ class k ,”-” means when $x_i < th$ then $x \in$ class k .

In this paper, to simplify discussion, we adopt models which are given a threshold at constant intervals Δ. In this case, selecting a smaller value for Δ makes it possible to fit further training data. Namely, we can adjust the model complexity by using Δ. In the following section, we call Δ the complexity parameter.

The decision stump enables us to create weak classifiers directly from training data. This means that the decision stumps are effective for practical applications. Hence, the model selection method of AdaBoost with the decision stumps has great importance for applications to real problems.

4 Information Criteria for Model Selection

AIC (Akaike information criterion) and BIC (Bayesian information criterion) have consistently been used as good criteria to select an appropriate model from a group of models of varying complexity. AIC is derived as an unbiased estimator about the mean value of the KL divergence, by applying asymptotic normality to model parameters, and is given by the following formula:

$$\mathrm{AIC} = -2\sum_{i=1}^{N} \log p(\boldsymbol{x}_i;\hat{\theta}) + 2d,$$

where $\hat{\theta}, d, N$ are maximum likelihood estimator(MLE), dimension of parameters and the number of instances of training data, respectively.

On the other hand, BIC is calculated from the negative value of the Laplace approximation of posterior probability and is given by the following formula:

$$\mathrm{BIC} = -2\sum_{i=1}^{N} \log p(\boldsymbol{x}_i;\hat{\theta}) + d\log N,$$

where $\hat{\theta}$ is the MLE as well as AIC; however, this model selection is also applicable to Bayes estimation as MAP estimation. The difference between these criteria is only a compensation term for likelihood, and both model selections are carried out by selecting minimum values. Therefore, we see from the above formulas that BIC results in a larger penalty for model complexity than AIC. It is well known that these criteria yield strong results for variable selection in regression models. However, the following condition is required in order to apply these criteria.

Uniqueness of parameters: Regularity of the model is presupposed in the derivation of AIC. Consequently, parameters must correspond to those of the probability model in a one-to-one fashion.

Differential of $p(\boldsymbol{x};\theta)$: The estimation of $\hat{\theta}$ is by MLE or MAP estimation; these criteria are then derived using the following equation $\left.\frac{\partial p(\boldsymbol{x};\theta)}{\partial\theta}\right|_{\theta=\hat{\theta}} = 0$.

These are intrinsically necessary conditions for application of AIC or BIC. However, these criteria have also been applied to singular models such as neural networks, which do not satisfy the above conditions, and even for these situations it has been reported that good results were obtained.

The parameters of AdaBoost using the decision stumps are given by coefficients of linear combinations of weak classifiers and the probability distribution is described by

$$p(y|\boldsymbol{x};\boldsymbol{\lambda}) = \frac{1}{Z}\exp\left\{f_{\boldsymbol{\lambda}}(\boldsymbol{x},y)\right\} \;\; (Z : \text{normalizing constant})\,.$$

In this case, model parameters correspond one-to-one to probability distributions by normalization of the probability model[1].

[1] If we are considering linear combinations of n weak classifiers, the model space can be identified with part of real projective space P^{n-1}D.

On the other hand, AdaBoost decreases successively the KL divergence with respect to the empirical distribution. This means that we can consider a probability model with sufficient learning times as an approximation of the MLE. The differential $\frac{\partial p(\boldsymbol{x};\theta)}{\partial \theta}$ therefore becomes very close to 0. In this paper, when the change of risk function $L(f_{\boldsymbol{\lambda}})$ becomes less than 1.0×10^{-5}, we consider the classifier obtained by AdaBoost as an approximation of the MLE. Strictly speaking, $\frac{\partial p(\boldsymbol{x};\theta)}{\partial \theta} \neq 0$ but when the above condition is satisfied, we apply AIC and BIC to AdaBoost and study their performance.

5 Dimension of Parameters

The weak classifiers utilized for the final probability distribution are usually a very small subset of the entire set of weak classifiers. For this reason, we cannot regard the number of weak classifiers as the dimension of the model space. On the other hand, there are many studies about the model selection problem of suitably choosing k parameters from $n\,(k \leq n)$ parameters. A decisive solution has not yet been obtained. However, as an example, the method which takes the minimum AIC model among k parameter models as the representation of all k parameters models is generally accepted. This method is based on the notion that the largest likelihood model is best among a set of models all having the same number of parameters, and the information criterion is applied only to models which have a relation of inclusion. If the models have a relation of inclusion, it is known that model selection by AIC can be regarded as model selection based on the likelihood ratio test using significance levels concerning the model's dimension. This fact shows the relevance of the above notion.

Complexity of models is described by $\varDelta$ in this paper. If the complexities of two models have the relation $\varDelta_1 = k\varDelta_2, (k \in \mathbb{Z}+)$ and the sets of models which have these complexities are denoted by $\mathcal{M}_{\varDelta_1}, \mathcal{M}_{\varDelta_2}$, then these two sets of models have the relation of inclusion, and therefore: $\mathcal{M}_{\varDelta_2} \subset \mathcal{M}_{\varDelta_1}$. We can also give meaning to an intermediate complexity of such models for $M_{\varDelta_3}$, with complexities related by $\varDelta_2 < \varDelta_3 < \varDelta_1$.

The AdaBoost algorithm determines the N weak classifiers which comprise the final classifier. These N classifiers are selected to minimize the KL divergence between the predictive distribution and empirical distribution, so we regard this final classifier as the largest likelihood model among all of the classifiers composed as linear combinations of N weak classifiers. From the above discussion, we suggest adjusting the complexity of the probability model by varying $\varDelta$ and selecting a suitable model by using the number of weak classifiers of the final classifier as the dimension of parameters. We use, under the above assumptions, notations AIC_{boost},BIC_{boost} as AIC and BIC for AdaBoost in the following sections.

6 Experiment

We give the Gaussian mixture model $p(x, y) = \sum_{i=1}^{3} a_i \phi_i(x, y; \boldsymbol{\mu}_i, \varSigma_i)$ as the true distribution. In addition, we suppose that each normal distribution

$\phi_i(x, y; \boldsymbol{\mu}_i, \Sigma_i)$ determines a class and that data x are generated from each normal distribution. We set the coefficients above as follows:

$$a_1 = 0.3, \;\; a_2 = 0.5, \;\; a_3 = 0.2 \; ,$$

$$\boldsymbol{\mu}_1 = \begin{pmatrix} 0 \\ 3 \end{pmatrix}, \;\; \boldsymbol{\mu}_1 = \begin{pmatrix} 3 \\ 0 \end{pmatrix}, \;\; \boldsymbol{\mu}_1 = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \Sigma_1 = \begin{pmatrix} 1 \, 0 \\ 0 \, 1 \end{pmatrix}, \Sigma_2 = \begin{pmatrix} 2 \, 0 \\ 0 \, 1 \end{pmatrix}, \Sigma_3 = \begin{pmatrix} 1 \, 0 \\ 0 \, 1 \end{pmatrix}.$$

We studied the behavior of information criteria for various complexities Δ with $n = 500$ training data generated from the above distribution.

The KL divergence and misclassification rate are calculated using 10000 test data in the following.

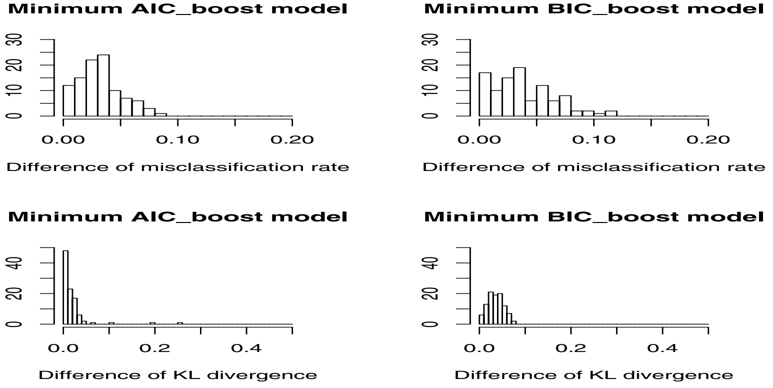


Fig. 2. Distribution of misclassification rate and KL Divergence ($n = 500$)

The upper row of Fig.2 shows the average difference in misclassification rate from 100 experiments between the model selected by AIC_{boost},BIC_{boost} and the actual minimum misclassification model, and the lower row shows a similar distribution about the KL divergence. We can see from Fig.2 that AIC_{boost} results in better estimation than BIC_{boost} with respect to the misclassification rate and yields a closer model to the true distribution than BIC_{boost} with KL divergence. However we also find that AIC_{boost} has selected a large KL divergence model in two experiments. This result illustrates that BIC_{boost} is superior to AIC_{boost} in respect of stability. KL divergence is considered to be a more accurate estimator than the misclassification rate; therefore the smallest misclassification rate classifier is not generally the best with respect to KL divergence. These results mean that we can observe in more detail the instability of AIC_{boost} through the KL divergence.

The relations between the number of instances of training data and the misclassification rate or the KL divergence are shown in Fig.3. The data from the most complicated model ($\Delta = 0.1$) are inserted as a reference. We can find the efficacy of AIC_{boost} and BIC_{boost} by comparison with the most complicated

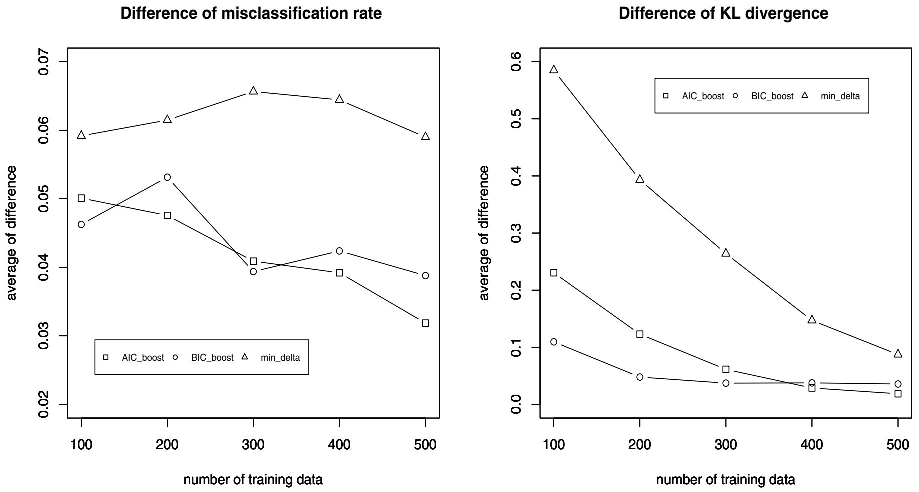


Fig. 3. Relation between the number of instances of training data and accuracy

model. Furthermore, Fig.3 illustrates that AIC$_{boost}$ chooses on the average a better model than BIC$_{boost}$ for large training datasets; in contrast, BIC$_{boost}$ yields a more stable result for small training datasets.

7 Discussion

The relations between the average number of weak classifiers adopted by AdaBoost and complexity Δ with respect to the number of instances of training data are plotted in Fig.4. We see that the number of weak classifiers increases

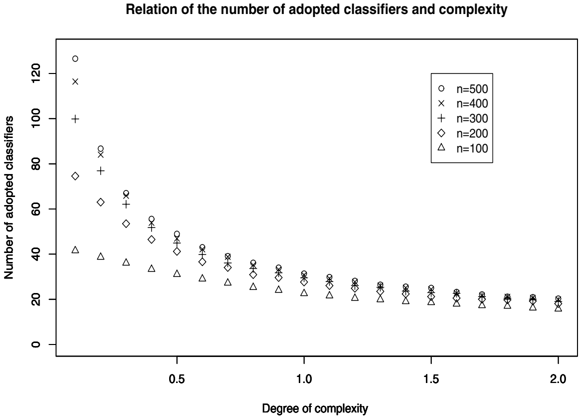


Fig. 4. Relation between the number of weak classifiers and complexity

exponentially as the complexity increases. This result indicates that choosing a smaller interval for thresholds as the complexity parameter goes down enables selection of a more detailed model. At the same time, the number of weak classifiers increases logarithmically as the amount of training data increases. Here, we conjecture that the number of weak classifiers should converge to the number of weak classifiers which are needed to approximate the true distribution.

In line with this, we found that the model dimension used in this paper is applied to the AIC$_{boost}$ and BIC$_{boost}$ after adjusting the penalty term exponentially with respect to complexity and adjusting logarithmically for the number of instances of training data.

8 Conclusion

We introduced the threshold interval Δ of weak classifiers as a model complexity parameter for AdaBoost using the decision stumps and examined the possibility of model selection. In addition, we proposed formal information criteria AIC$_{boost}$ and BIC$_{boost}$ using the number of weak classifiers adopted by the AdaBoost as the dimension of parameters. We also investigated their efficiencies and characteristics experimentally. Giving theoretical meaning to the results we have obtained is a topic for future research.

Acknowledgment

This work was supported by the Ministry of Education, Science, Sports, and Culture in Japan, Grand-in-aid for scientific research 18079007.

References

1. Amari, S., Nagaoka, H.: Method of information geometry. American Mathematical Society and Oxford Press (2000)
2. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of online learning and an application to boosting. Journal of Computing Systems and Science 55(1), 119–139 (1997)
3. Freund, Y.: Experiments with a new boosting algorithm. In: Proc. 13th International Conference on Machine Learning Bari, pp. 146–148 (1996)
4. Hastie, T., Tibshirani, R., Friedman: The Elements of Statistical Learning:Data Mining Inferecnce and Prediction. Springer, Heidelberg (2001)
5. Murata, N., Takenouchi, T., Kanamori, T., Eguchi, S.: Information geometry of U-Boost and Bregman divergence. Neural Computation 16(7), 1437–1481 (2004)
6. Murata, N., Yoshizawa, S., Amari, S.: Network information criterion-determining the number of hiddenunits. IEEE Transactions on Neural Networks 5(6), 865–872 (1994)
7. Rätsch, G., Onoda, T., Mueller, K.-R.: Soft margin for AdaBoost. Machine Learning 42(3), 287–320 (2001)
8. Schapire, R.E.: Boosting the Margin: A new explanation for the effectiveness of voting methods. Annals of Statistics 26(5), 1651–1686 (1998)

The Diversity of Regression Ensembles Combining Bagging and Random Subspace Method

Alexandra Scherbart and Tim W. Nattkemper

Biodata Mining & Applied Neuroinformatics Group, Faculty of Technology,
Bielefeld University, Postfach 10 01 31, 33501 Bielefeld
{ascherba,tnattkem}@techfak.uni-bielefeld.de

Abstract. The concept of Ensemble Learning has been shown to increase predictive power over single base learners. Given the bias-variance-covariance decomposition, diversity is characteristic factor, since ensemble error decreases as diversity increases. In this study, we apply Bagging and Random Subspace Method (RSM) to ensembles of Local Linear Map (LLM)-type, which achieve non-linearity through local linear approximation, supplied with different vector quantization algorithms. The results are compared for several benchmark data sets to those of RandomForest and neural networks. We can show which parameters are of major influence on diversity in ensembles and that using our proposed method of LLM combining RSM we are able to achieve results obtained by other reference ensemble architectures.

1 Introduction

Ensemble Learning was paid much attention in the literature recently. The output of several learning algorithms is combined to build powerful learning machines. This concept has been proven to increase predictive power over single base learners for a great variety of classification and regression problems. Given a set of observations $T = \{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\}$, the problem in regression ensembles is to select a set of appropriate predictors $H = \{f_1, \ldots, f_M\}$ from the base hypothesis space $\mathcal{H}$ and to aggregate the outputs of the M predictors to one ensemble vote:

$$F(\mathbf{x}) = \sum_{m=1}^{M} w_m f_m(\mathbf{x}), \text{ with } \sum_m w_m = 1,$$

where F is given as a convex combination of the component estimators.

Krogh et al. proved in [1] that the ensemble error is guaranteed to be less than or equal to the average quadratic error of the individual components. The quadratic error of the ensemble $\bar{e}$ can be decomposed into two terms similar to the bias-variance decomposition for one single learner. To capture the covariance

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 911–918, 2009.

in ensembles, one additional term is introduced leading to the bias-variance-covariance decomposition [2]:

$$\bar{e} = (F - \mathbf{y})^2 = \sum_{m=1}^{M} w_m(\mathbf{y} - f_m)^2 - \sum_{m=1}^{M} w_m(f_m - F)^2.$$

The first term is the average error of individual predictors, while the second measures how much each single ensemble member diverges from the ensemble output F, the so called diversity. For a given fixed mean error of individual predictors, the ensemble error rate is reduced when increasing the diversity. This leads to two potentially conflicting targets of minimizing the error of each ensemble member and maximizing the diversity of the ensemble. In the following, we give a short introduction in the sources of diversity:

Bagging One well-studied and intuitive way of combining the component predictors is simple averaging ($w_m = 1/M$). The method of Bagging was introduced by Breiman [3]. He proposed aggregating a set of predictors generated from bootstrapped samples $T_1, \ldots, T_m$, randomly selected N patterns with replacement from the original set of N training patterns.

From the training set T, M bootstrap training sets T_m are generated and the constructed regressors $f(\mathbf{x}, T_m) = f_m(\mathbf{x})$ are used to form the bagged predictor F. The bootstrap training sets T_m contain about a fraction $(1 - 1/M)^M \approx 0.63$ of the original N instances. Therefore about one-third of the patterns are left out in every bootstrapped sample and are called "out-of-bag" (OOB) data. The out-of-bag estimates are given by aggregation over the regressors f_m which do not contain pattern $(\mathbf{x}, y) \in T$. This OOB estimates tend to overestimate the test error rate for the prediction is based on only one-third of the M regressors. The error rate decreases with an increasing number of regressors in ensembles.

Random Subspace Method Following [4], one way to manipulate the set of hypotheses accessible to a learner is to supply each learner with a slightly altered different training set to generate diverse ensemble members. While Bagging is one implicit method to build altered learners, other resampling methods supply each member with all N patterns, but with different subset of variables. The Random Subspace Method (RSM) [5] was proposed for classifier of multiple trees constructed in randomly chosen subspaces.

A few studies exist, which propose to combine Bagging with RSM in regression context. When supplying the resampling methods, it is expected that every single predictor is to adapt to different parts of the same learning task. In our study, we apply a regression ensemble, namely the Local Linear Map [6], a type of artificial neural net, supplied with different vector quantization (VQ) algorithms. It combines an unsupervised VQ with supervised linear learning principles. The LLM can learn global non-linear regression functions by fitting a set of *local* linear functions to the training data. It is very efficient (i.e. fast training and adaptation to addition data, and low memory-usage) and offers transparency.

2 Combining Bagging with RSM in LLM Ensembles

Local Linear Map: Motivated by the Self-Organizing Maps (SOM) [7], an LLM consists of a set of n_l regular ordered nodes $\mathbf{v}_i, i = 1, \ldots, n_l$, which are connected to each other via a two-dimensional grid structure, defining a neighborhood between the nodes and a topology in feature space. Each node consists of a triple $\mathbf{v}_i = (\mathbf{w}_i^{\text{in}}, \mathbf{w}_i^{\text{out}}, \mathbf{A}_i)$. The vectors $\mathbf{w}_i^{\text{in}} \in \mathbb{R}^{d_{\text{in}}}$ are used to build prototype vectors adapting to the statistical properties of the input data $\mathbf{x}_\xi \in \mathbb{R}^{d_{\text{in}}}$. The vectors $\mathbf{w}_i^{\text{out}} \in \mathbb{R}^{d_{\text{out}}}$ approximate the distribution of the target values $\mathbf{y}_\xi \in \mathbb{R}^{d_{\text{out}}}$. The matrices $\mathbf{A}_i \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ are locally trained linear maps from the input to the output space. In the unsupervised training phase, the prototype vectors $\mathbf{w}_i^{\text{in}}$ are adapted following the SOM learning rule: the vectors $\mathbf{w}_i^{\text{in}}$ are pulled towards the input pattern $\mathbf{x}_\xi$ according to the distance between the input pattern and the corresponding closest prototype in input space $\mathbf{w}_\kappa^{\text{in}}$, with $\kappa = \operatorname*{argmin}_i \left\{\|\mathbf{x}_\xi - \mathbf{w}_i^{\text{in}}\|\right\}$. After unsupervised adaptation and tessellation of the input space, an input feature vector is mapped to an output by the corresponding local expert: $\mathcal{C}(\mathbf{x}) = \mathbf{w}_\kappa^{\text{out}} + \mathbf{A}_\kappa \left(\mathbf{x}_\xi - \mathbf{w}_\kappa^{\text{in}}\right)$. The weights $\mathbf{w}_i^{\text{out}}$ and the linear map $\mathbf{A}_i$ are changed iteratively by the gradient descent learning rules.

We also applied Growing Neural Gas (GNG) [8], Neural Gas (NG) [9] and Fuzzy C-means (FCL) [10] clustering instead of SOM in the input space for comparison. The concept of approximating nonlinear functions by fitting simple models to localized subsets of the data is related to other regression approaches like Locally-Weighted Regression [11] and to radial basis functions [12].

We propose an ensembles architecture supplying Local Linear Maps, namely the **2DELL** (**2-d**im. **E**nsemble **l**earnining with **L**LM):

Algorithm 1. 2DELL(M=100, k=d, VQ(”SOM”, 2 × 5))

```
Require: Training set $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ with d variables
Require: M, the number of predictors in the ensemble
Require: k, the size of random subspaces (k<d)
  for $m = 1$ to $M$ do
    Create bootstrapped samples $T_m$ with replacement of size N from training data $T$
    Take random subset of variables of size $k$ (without replacement)
    Train LLM model $f_m$ with training data of size $N \times k$
    Determine mean error (MSE) for each ensemble member:
        $MSE_m = \frac{1}{N} \sum_n (f_m(\mathbf{x}_n) - y_n)^2, \text{ where } (\mathbf{x}_n, y_n) \in T_m$
  end for
  Calculate OOB estimates by:
    $MSE_{oob} = \frac{1}{N} \sum_n \left( \sum_{i \in S_{mn}} \frac{1}{|S_{mn}|} f_i(\mathbf{x}_n) - y_n \right)^2, \text{ with } S_{mn} = \{m | (\mathbf{x}_n, y_n) \notin T_m\}$
```

3 Empirical Results

We examined which parameters influence the diversity in regression tasks in the context of Local Linear Maps. Parameters evaluated are: Local Linear Map underlying VQ-algorithm (GNG, SM, NG and FCL), number of variables $k \in 4 : max$, number of maps $M \in \{1, 2, 10, 25, 100\}$. We then compare the resulting generalization performance of the proposed ensemble to Random Forests and MLP-based ensembles.

Datasets. In our study, we use five different benchmark data sets, artificial as well as real-world ones to allow a general comparison to other regression and ensemble architectures applied to these problems.

1. Friedman#1: This artificial data set has ten independent variables uniformly distributed over [0,1], while only five out of these ten are used to define y. $y = 10 \sin(\pi \cdot x_1 \cdot x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + e,$ where e is $N(0, 1)$.
2. Friedman#2: Second artificial data set corresponds to input vectors uniformly distributed over the ranges $0 \leq x_1 \leq 100$, $40\pi \leq x_2 \leq 560\pi$, $0 \leq x_3 \leq 1$, $1 \leq x_4 \leq 11$: $y = (x_1^2 + (x_2 \cdot x_3 - (1/(x_2 \cdot x_4)))^2)^{0.5} + e,$ where e is $N(0, 125)$.
3. Friedman#3: Inputs are 4 independent variables uniformly distrtibuted as above but the outputs are generated according to: $y = \arctan\left((x_2 \cdot x_3 - (x_2 \cdot x_4)^{-2}) \cdot x_1^{-1}\right) + e,$ where e is $N(0, 0.1)$.
4. NO2: The data are a subsample of 506 observations from a data set with 7 variables, that originate in a study where air pollution at a road is related to traffic volume and meteorological variables, collected by the Norwegian Public Roads Administration [13].
5. AAindex: This in-house data set is one small, noisy, high-dimensional data set from mass spectrometry (MALDI mass spectra) with 372 peptides. The feature vectors are built from the peptide sequences by physico-chemical information about the amino acids constituting the peptide. The feature vectors are attributes taken from the amino acid index database [14](aaindex) extended by peptide length, mass, and numbers and fractions of acidic, basic, polar, aliphatic and arginine residues, yielding 531-dimensional vectors.

All data sets are of small size and low-dimensional, except for the last one, AAindex, where the dimensionality even exceeds the number of available patterns. We put aside randomly selected 10% of the input patterns for testing and the remaining 90% are used to build the predictors. Tab. 1 gives on overview on the studied data sets. The error rates are averaged over 100 repetitions. All variables are centered and normalized prior to training.

3.1 Ensemble Regression Results

We are interested in further insights into the interdependencies between the evaluated parameters of our proposed LLM-architecture combining Bagging and RSM and resulting diversity. It is worth noting, that the ensemble error rate

Table 1. Data set survey

Data set	Friedman#1	Friedman#2	Friedman#3	NO2	AAindex
# variables	10	4	4	7	531
# Training	200	200	200	450	372
# Test	2000	2000	2000	10%	10%

$\bar{e}$ is given by the difference between mean error per map and diversity, if not explicitly mentioned. We denote this by $\bar{e} = (F - \mathbf{y}^2) = MSE_{maps} - Div$, where $MSE_{maps} = \sum_m 1/M(\mathbf{y} - f_m)^2$. The results are measured concerning MSE_{maps} and diversity Div.

Diversity vs the size of networks. First we varied the number of nodes or the size of the network when the other parameters are kept fixed. The default call to 2DELL() is done by $(M = 100, k = d, VQ(.))$. For VQ $\in$ NG or FCL, we tried reasonable network sizes $2 \leq n_l \leq 15$, while two dimensional SOM-grid is expanded from $(1, 2)$ up to $(5, 7)$. The optimal network size is determined by the best test error rate over 10 iterations.

Increasing size of networks leads to loss in generalization performance, since every net gets too specialized to the presented subtask. As expected, powerful ensembles are generated with small nets. Change in test error rate is quite stable for small sizes (3 up to $\approx$ 10). At least for the small data sets, roughly speaking, this means that the gain in diversity is abrogated by the increase in MSE_{maps}.

Diversity vs the size of ensemble (M). When varying the size of ensemble, the parameter M is in focus. Given the previously determined optimal number of nodes for each type of VQ algorithm, they were kept fixed as constants. We analyzed the error rates of ensembles of size $M \in \{1, 2, 10, 25, 100\}$. In Fig. 1 the mean error rate per map MSE_{maps} is plotted against the corresponding diversity Div for Friedman#1 data set. The first two figures on the right give an overview over the two mentioned measures, when varying the number of ensemble NG or SOM members. The very left points correspond to results when all ten $(k = d)$ variables are supplied to training (k indicated by small digit nearby). Apart from a small spread in MSE_{maps}, the data points yield a vertical trajectory. With an increasing number of ensemble members, the mean OOB and test error decrease. Therefore, the decrease in error rate can be traced back only to the gain in diversity.

Combining Bagging and Random Subspace method. We evaluate the change of diversity and ensemble error rate when applying RSM. Randomly selected variables uniformly distributed build up the feature space. Every ensemble member is generated with only a *subset of training patterns and supplying only a subset of variables.* Every component is expected to adapt to different parts of the same learning task. The evaluation of the 2DELL varying over the parameters k and M yields the following best results given in Tab. 2. For comparison purposes, the resulting best performances of RandomForests (RF) [15] are mentioned, and additionally the results of artificial neural network of MLP-type with 5 hidden nodes.

Table 2. Test error (MSE), averaged over 100 iterations. The results for Friedman#2 are given in units of 10^{+3}, while those for Friedman#3 in unit of 10^{-3}. * Results are taken from [15]. ** The MLP results are given by $M = 5$ with 5 hidden nodes.

	GNG	NG	SOM	FCL	RF	ANN**
Friedman#1	6.19	5.71	**5.47**	7.19	5.7*	6.26
Friedman#2	21.9	19.3	**18.9**	36.6	19.6*	191.7
Friedman#3	41.5	30.9	27.6	54.0	21.6*	**19.6**
NO2	0.287	0.247	0.239	0.267	**0.216**	0.277
AAindex	1.86	1.05	**1.01**	1.55	1.26	1.48

In case of Friedman#1, Friedman#2 and AAindex the proposed SOM-LLM based regression ensembles outperform every other applied method. For the other data sets the 2DELL with SOM as VQ-algorithm performs second (third) best. From Tab. 2 it can be observed that SOM-LLM ensembles yield the best generalization performance compared to every other VQ-algorithm.

In Fig.1 (left column) the OOB error estimate, the test error and the diversity is given for data set Friedman#1 for varied size k of Random Subspace. The OOB error rate overstimates the test error. Both are decreasing rapidly when enlarging the number of randomly selected variables. Diversity is maximum if $k = 6$ with SOM-LLM and decreases nearly linear if $k > 6$.

Fig.1 (right column) accounts for different ensembles sizes M and at the same time varying number of variables k for RSM. For this purpose, the MSE_{maps} is plotted against the diversity ($Div = MSE_{maps} - \bar{e}$). Every point represents one combination of (M, k). Points are connected to each other if they belong to the same ensemble size. The connected lines can be seen as one trajectory through hypotheses space.

While the components diversity and MSE_{maps} should be balanced against each other, the points minimizing error rate and also maximizing diversity are optimal in that sense, and build a so called pareto-front. The pareto-front is shifted continuous against an optimum, towards the upper left corner, until a local maximum is reached for $M = 100$. In case of NG as well as SOM, from the perspective of Div and MSE_{maps}, the point $(M = 100, k = 9)$ is optimal. Varying the number of variables selected, the gradient of Div and MSE_{maps} is greater 1, since the diversity decreases more slowly than MSE_{maps} in this region. Applying Random Subspace method with (k=9) yields better generalization performance than without (k=10).

Comparison of learning algorithms encouraging diversity. Neural Gas and SOM show similar behaviour in terms of error rates, performance and diversity. FCL and GNG in general represent other groups regarding this context. While FCL shows small MSE_{maps} and low diversity, the loss in generalization performance with GNG may be due an overadaptation to the presented resampled subtask yielding high average error rate with high variance, though striking diversity. For hybrid ensembles, set of applied VQ algorithms should be combined with respect to similar MSE_{maps}-diversity behaviour.

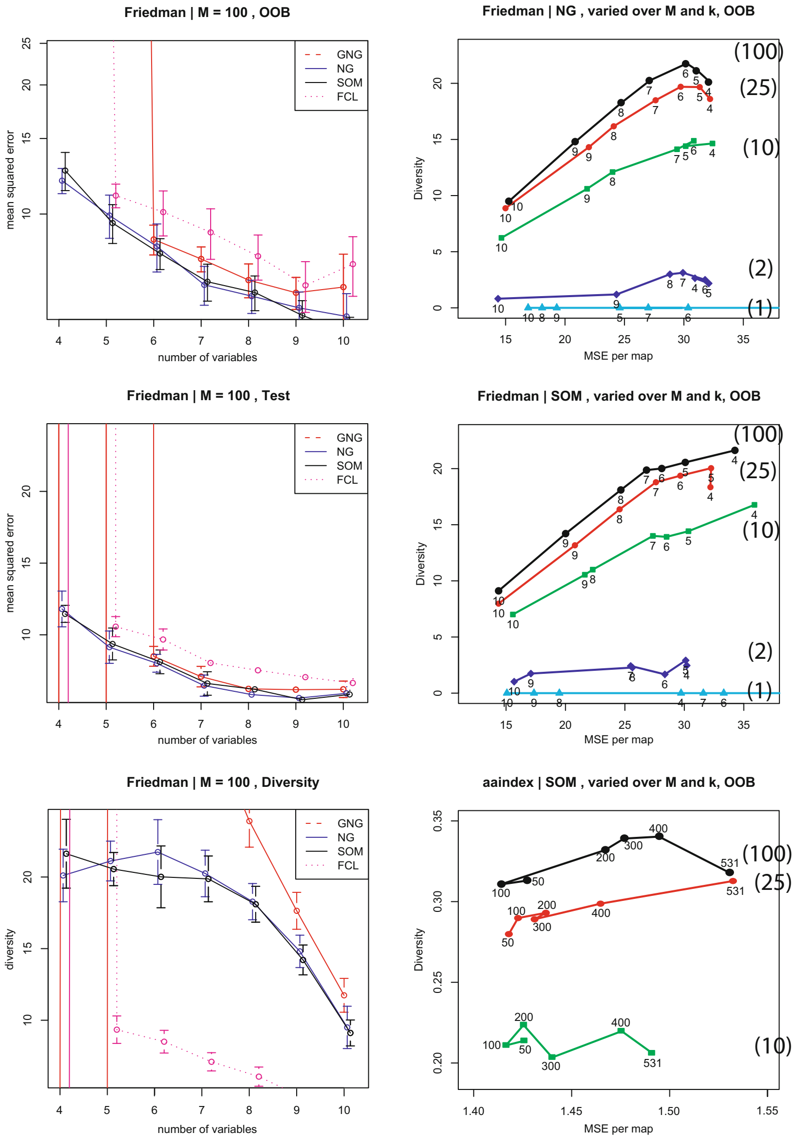


Fig. 1. (Left) OOB, test error rate and diversity for Friedman#1 data set when varying over the number of variables. (Right) Plot MSE per map (MSE_{maps}) against diversity for Friedman#1 in case of NG(a), SOM(b), and AAindex data set(c). Every point represents one combination of (M, k). Points are connected to each other if they belong to the same ensemble size (M).

4 Conclusion

In Ensemble Learning, the output of several learning algorithms is aggregated to build powerful learning machines. This concept has been proven to increase predictive power over single base learners. We proposed a regression ensemble learning with combining resampling methods Bagging and Random Subspace Method and evaluation from a perspective of diversity. The proposed Local Linear Map (LLM) Ensemble is applied to different benchmark data sets and compared to Random Forest and ANNs regarding generalization performance with comparable results. LLM ensembles were found to be sufficient, appropriate predictors. We evaluated the change in diversity when varying the parameters of ensemble and show that powerful ensembles are generated with small nets. We demonstrated the usefulness of combining Bagging with Random Subspace in terms of diversity. The factor mostly increasing diversity and predictive power is found to be the supplied VQ-method, among the number of variables building Random Subspaces, followed by number of ensemble members.

References

1. Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation, and active learning. In: Adv. in NIPS, pp. 231–238. MIT Press, Cambridge (1995)
2. Brown, G.: Diversity in neural network ensembles. Technical report (2004)
3. Breiman, L.: Bagging predictors. Machine Learning, 123–140 (1996)
4. Brown, G., Wyatt, J., Harris, R., Yao, X.: Diversity creation methods: A survey and categorisation. Journal of Information Fusion 6, 5–20 (2005)
5. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Trans. on Pattern Analysis and Machine Intelligence 20, 832–844 (1998)
6. Ritter, H.: Learning with the self-organizing map. In: Kohonen, T., et al. (eds.) Artificial Neural Networks, pp. 379–384. Elsevier Science Publishers, Amsterdam (1991)
7. Kohonen, T.: Self-organized formation of topologically correct feature maps. Biological Cybernetics 43, 59–69 (1982)
8. Fritzke, B.: Fast learning with incremental RBF networks. Neural Processing Letters, 2–5 (1994)
9. Martinetz, T.M., Berkovich, S.G., Schulten, K.J.: 'Neural Gas' network for vector quantization and its application to time-series prediction. IEEE Trans. Neural Networks. 4, 558–569 (1993)
10. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Kluwer Academic Publishers, Norwell (1981)
11. Cleveland, W.S., Devlin, S.J.: Locally-weighted regression: An approach to regression analysis by local fitting. J. of the American Stat. Assoc. 83, 596–610 (1988)
12. Millington, P.J., Baker, W.L.: Associative reinforcement learning for optimal control. In: Proc. Conf. on AIAA Guid. Nav. and Cont., vol. 2, pp. 1120–1128 (1990)
13. Vlachos, P.: Statlib datasets archive (2005)
14. Kawashima, S., Ogata, H., Kanehisa, M.: AAindex: Amino Acid Index Database. Nucleic Acids Res. 27(1), 368–369 (1999)
15. Breiman, L.: Random forests. Machine Learning, 5–32 (2001)

On Weight-Noise-Injection Training

Kevin Ho[1], Chi-sing Leung[2], and John Sum[3,⋆]

[1] Department of Computer Science and Communication Engineering,
Providence University, Sha-Lu, Taiwan
ho@pu.edu.tw

[2] Department of Electronic Engineering, City University of Hong Kong
Kowloon Tong, KLN, Hong Kong
eeleungc@cityu.edu.hk

[3] Institute of E-Commerce, National Chung Hsing University
Taichung 402, Taiwan
pfsum@nchu.edu.tw

Abstract. While injecting weight noise during training has been proposed for more than a decade to improve the convergence, generalization and fault tolerance of a neural network, not much theoretical work has been done to its convergence proof and the objective function that it is minimizing. By applying the Gladyshev Theorem, it is shown that the convergence of injecting weight noise during training an RBF network is almost sure. Besides, the corresponding objective function is essentially the mean square errors (MSE). This objective function indicates that injecting weight noise during training an radial basis function (RBF) network is not able to improve fault tolerance. Despite this technique has been effectively applied to multilayer perceptron, further analysis on the expected update equation of training MLP with weight noise injection is presented. The performance difference between these two models by applying weight injection is discussed.

1 Introduction

Many methods have been developed throughout the last two decades to improve the fault tolerance of a neural network. Well known methods include injecting random fault during training [25,5], introducing network redundancy [23], applying weight decay learning [9], formulating the training algorithm as a nonlinear constraint optimization problem [10,22], bounding weight magnitude during training [7,15,17], and adding fault tolerant regularizer [2,19,27]. A complete survey on fault tolerant learning methods is exhaustive. Readers please refer to [8] and [29] for reference.

Amongst all, the fault-injection-based on-line learning algorithms are of least theoretical studied. By fault injection, either fault or noise is introduced to a neural network model before each step of training. This fault could either be node fault (stuck-at-zero), weight noise or input noise. As many studies have been reported in the literature on input noise injection [1,4,24,13,14], the primary

[⋆] Corresponding author.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 919–926, 2009.

focus of this paper is on weight noise injection. Our companion paper [28] will be focus on node fault injection.

Suppose a neural network consists of M weights. Let $\theta \in R^M$ be the weight vector of a neural network model and the update equation is given by $\theta(t+1) = \theta(t) - F(x(t+1), y(t+1), \theta(t))$. The idea of weight noise injection is to replace $\theta(t)$ in the factor $F(\cdot, \cdot, \theta(t))$ by $F(\cdot, \cdot, \tilde{\theta}(t))$. Here the elements of $\tilde{\theta}(t)$ is of the form $\tilde{\theta}_i(t) = \theta_i(t) + \Delta\theta_i(t)$. The factor $\Delta\theta_i(t)$ is the weight noise injected. The update equation is thus defined as follows :

$$\theta(t+1) = \theta(t) - F(x(t+1), y(t+1), \tilde{\theta}(t)). \tag{1}$$

Despite injecting weight noise to improve convergence ability, generalization and fault tolerance have long been investigated [20,21,16,11] for MLPs and recurrent neural networks, and theoretical analysis on applying such technique to MLP has been reported [1], little is known about the effect of injecting weight noise during training an RBF network.

In this paper, an analysis on weight-noise-injection-based training will be presented. In the next section, the convergence proof and the objective function of RBF training with on-line weight injection will be analyzed. Section 3 will show the analysis on the case of MLP. The conclusion will be presented in Section 4.

2 RBF Training with Weight Noise Injection

2.1 Network Model

Let $\mathcal{M}_0$ be an unknown system to be modeled. The input and output of $\mathcal{M}_0$ are denoted by x and y respectively. The only information we know about $\mathcal{M}_0$ is a set of measurement data $\mathcal{D}$, where $\mathcal{D} = \{(x_k, y_k)\}_{k=1}^N$. Making use of this data set, an estimated model $\hat{\mathcal{M}}$ that is *good* enough to capture the *general behavior* of the unknown system can be obtained.

For $k = 1, 2, \cdots, N$, we assume that the true model is governed by an unknown deterministic system $f(x)$ together with mean zero Gaussian output noise :

$$\mathcal{M}_0 \quad : \quad y_k = f(x_k) + e_k, \tag{2}$$

Besides, we assume that the unknown system $f(x)$ can be realized by an RBF network consisting of M hidden nodes, i.e.

$$y_k = \sum_{i=1}^{M} \theta_i^* \phi_i(x_k) + e_k \tag{3}$$

for all $k = 1, 2, \cdots, N$ and $\phi_i(x)$ for all $i = 1, 2, \cdots, M$ are the radial basis functions given by $\phi_i(x) = \exp\left(-\frac{\|x-c_i\|^2}{\sigma}\right)$, where c_is are the centers of the radial basis function and the positive parameter $\sigma > 0$ controls the width of the radial basis functions. In vector form, Equation (3) can be rewritten as follows :

$$y_k = \phi(x_k)^T \theta^* + e_k, \tag{4}$$

where $\phi(\cdot) = (\phi_1(\cdot), \phi_2(\cdot), \cdots, \phi_M(\cdot))^T$ and $\theta^* = (\theta_1^*, \theta_2^*, \cdots, \theta_M^*)^T$.

2.2 Weight Noise Injection Training

While a network is trained by the idea of weight noise injection, the update equation will be given by

$$\theta(t+1) = \theta(t) + \mu_t(y_t - \phi^T(x_t)\tilde{\theta}(t))\phi(x_t), \tag{5}$$

where μ_t is (for $t \geq 1$) the step size at the t^{th} iteration,

$$\tilde{\theta}_i(t) = \begin{cases} \theta_i(t) + \beta_i & \text{for additive noise injection,} \\ \theta_i(t) + \beta_i\theta_i(t) & \text{for multiplicative noise injection.} \end{cases} \tag{6}$$

β_i for all $i = 1, 2, \cdots, M$ are independent mean zero Gaussian noise with variance S_β. Normally, it is assumed that the value of S_β is small. Although the theoretical proof presented later in this paper applies to any bounded value, it is meaningless to consider a large value of S_β.

2.3 Convergence and Objective Function

Theory of stochastic approximation has been developed for more than half a century for the analysis of recursive algorithms. Advanced theoretical works for complicated recursive algorithms have still been under investigation [18]. The theorem applied in this paper is based on Gladyshev Theorem [12].

Theorem 1 (Gladyshev Theorem [12]). *Let $\theta(t)$ and $M(\theta(t), \omega(t))$ for all $t = 0, 1, 2,$ and so on be m-vectors. $\omega(t)$ for all $t = 0, 1, 2,$ and so on are i.i.d. random vectors with probability density function $P(\omega)$[1]. Consider a recursive algorithm defined as follows :*

$$\theta(t+1) = \theta(t) - \mu_t M(\theta(t), \omega(t)). \tag{7}$$

In which, the expectation of $M(\theta, \omega)$ over ω,

$$\bar{M}(\theta) = \int M(\theta, \omega)P(\omega)d\omega, \tag{8}$$

has unique solution θ^* such that $\bar{M}(\theta^*) = 0$.

Suppose there exists positive constants κ_1 and κ_2 such that the following conditions are satisfied :

(C1) $\mu_t \geq 0$, $\sum_t \mu_t = \infty$ *and* $\sum_t \mu_t^2 < \infty$.
(C2) $\inf_{\varepsilon < \|\theta - \theta^*\| < \varepsilon^{-1}} (\theta - \theta^*)^T \bar{M}(\theta) > 0$, *for all* $\varepsilon > 0$.
(C3) $\int \|M(\theta, \omega)\|^2 P(\omega)d\omega \leq \kappa_1 + \kappa_2\|\theta\|^2$.

Then for $t \to \infty$, $\theta(t)$ converges to θ^* with probability one.

[1] In the following convergence proof, $\omega(t) = (x_t, y_t, \beta_t)$. Owing not to confuse the time index t with the element index k, the subscript t is omitted. So that $\omega(t) = (x_t, y_t, \beta)$.

Applying Gladyshev Theorem, the following theorem can be proved for injecting weight noise.

Theorem 2. *For injecting (additive or multiplicative) weight noise during training an RBF network, the weight vector $\theta(t)$ will converge with probability one to*

$$\theta^* = \left(\frac{1}{N} \sum_{k=1}^{N} \phi(x_k) \phi^T(x_k) \right)^{-1} \frac{1}{N} \sum_{k=1}^{N} y_k \phi(x_k). \tag{9}$$

Proof. For a RBF network that is trained by injecting multiplicative weight noise,

$$\theta(t+1) = \theta(t) + \mu_t (y_t - \phi^T(x_t)\tilde{\theta}(t))\phi(x_t), \tag{10}$$

$$\tilde{\theta}_i = (1+\beta_i)\theta_i, \quad \beta_i \sim \mathcal{N}(0, S_\beta), \ \forall\, i = 1, \cdots, M. \tag{11}$$

Suppose S_β is small. Taking expectation of the second term in right hand side of the first equation with respect to β, it can readily be shown that

$$\int_{\Omega_{\tilde{\theta}(t)}} (y_t - \phi^T(x_t)\tilde{\theta}(t))\phi(x_t) d\tilde{\theta}(t) = (y_t - \phi^T(x_t)\theta(t))\phi(x_t).$$

Further taking expectation of the above equation with respect to x_t and y_t, $h(\theta(t))$ will be given by

$$h(\theta(t)) = \frac{1}{N} \sum_{k=1}^{N} (y_k - \phi^T(x_k)\theta(t))\phi(x_k), \tag{12}$$

Therefore, the solution θ^* is $\theta^* = H_\phi^{-1} Y$.

Next, we are going to apply the Gladyshev Theorem for the convergence proof. Normally, the first condition can easily be satisfied. It is because the step size μ_t could be pre-defined. So, we skip the proof of Condition (C1) for simplicity.

To prove Condition (C2), we first note that $\bar{M}(\theta) = -h(\theta)$. We further let $Y = \frac{1}{N}\sum_{k=1}^{N} y_k \phi(x_k)$ and $\omega = (x_t, y_t, \beta)$. Hence, for all $\|\theta - \theta^*\| > 0$, we have $-(\theta - \theta^*)^T h(\theta) = -(\theta - \theta^*)^T (Y - H_\phi \theta)$, which is greater than zero.

To prove Condition ($C3$), we consider the Equation (10). By triangle inequality,

$$\|M(\theta,\omega)\|^2 \leq \|y_t \phi(x_t)\|^2 + \|\phi(x_t)\phi(x_t)^T \theta\|^2 + \|\phi(x_t)\phi(x_t)^T A_\theta \beta\|^2, \tag{13}$$

where $A_\theta = \mathbf{diag}\{\theta_1, \theta_2, \cdots, \theta_M\}$. Clearly, the first term in the RHS of the inequality is a factor independent of θ. We let it be $K(x_t, y_t)$ as before. The second term is $\theta^T(\phi(x_t)\phi(x_t)^T)^2\theta$. In which the matrix $(\phi(x_t)\phi(x_t)^T)^2$ is symmetric and of bounded elements. Therefore, its largest eigenvalue must also be a bounded nonnegative number, say $\lambda(x_t)$. Taking expectation of the third term with respect to β,

$$\begin{aligned} \|\phi(x_t)\phi(x_t)^T A_\theta \beta\|^2 &= S_\beta \sum_{i=1}^{M} \theta^2 \left(\phi(x_t)\phi(x_t)^T \phi(x_t)\phi(x_t)^T\right)_{ii}, \\ &\leq S_\beta \max_i \{(\phi(x_t)\phi(x_t)^T \phi(x_t)\phi(x_t)^T)_{ii}\} \|\theta\|^2. \end{aligned}$$

As a result,

$$\int \|M(\theta,\omega)\|P(\beta)d\beta \leq K(x_t,y_t) + S_\beta(\lambda(x_t)\max_i\{(\phi(x_t)\phi(x_t)^T)^2_{ii}\})\|\theta\|^2.$$

Further taking the expectation of the above inequality with respect to x_t and y_t, one can readily show that Condition ($C3$) can be satisfied and the proof is completed.

To prove the convergence of injecting additive weight noise, simply defining $\tilde{\theta}(t)$ in Equation (10) by $\theta(t)+\beta$ and A_θ in Equation (13) by an $M \times M$ identity matrix. It will be clearly that $h(\theta(t))$ will be identical to Equation (12) and the third term will be independent of θ. The proof of Condition (C3) will be accomplished. **Q.E.D.**

As the solution θ^*, by either injecting additive weight noise or multiplicative weight noise, is identical to the solution obtained by the ordinary pseudo-inverse, the following theorem can be implied.

Theorem 3. *The objective function of injecting (additive or multiplicative) weight noise during training an RBF is identical to the mean square errors.*

$$\mathcal{L}(\theta|\mathcal{D}) = \frac{1}{N}\sum_{k=1}^{N}(y_k - f(x_k,\theta))^2. \tag{14}$$

3 MLP Training with Weight Noise Injection

3.1 Injecting Multiplicative Weight Noise

Consider a nonlinear neural network $g(x,\theta)$, where both its gradient vector $g_\theta(x,\theta)$ and Hessian matrix $g_{\theta\theta}(x,\theta)$ exist. Similar to that of RBF learning, the online weight noise injection learning algorithm for $g(x,\theta)$ given a dataset $\mathcal{D} = \{(x_k,y_k)\}_{k=1}^N$ can be written as follows :

$$\theta(t+1) = \theta(t) + \mu_t(y_t - g(x_t,\tilde{\theta}(t)))g_\theta(x_t,\tilde{\theta}(t)). \tag{15}$$

$$\tilde{\theta}(t) = \theta(t) + A_\beta\theta(t). \tag{16}$$

Here, $A_\beta = \mathbf{diag}\{\beta_1,\beta_2,\cdots,\beta_M\}, \quad \beta_i \sim \mathcal{N}(0,S_\beta)$. For small S_β, one can assume that $\tilde{\theta}$ is close to θ and then apply Taylor expansion to $g(\cdot,\cdot)$ and $g_\theta(\cdot,\cdot)$ and get that

$$g(x_t,\tilde{\theta}(t)) \approx g(x_t,\theta(t)) + g_\theta(x_t,\theta(t))^T A_\beta\theta(t), \tag{17}$$

$$g_\theta(x_t,\tilde{\theta}(t)) \approx g_\theta(x_t,\theta(t)) + g_{\theta\theta}(x_t,\theta(t))A_\beta\theta(t). \tag{18}$$

Putting the above approximations into Equation (15) and taking expectation over β, it is readily shown that

$$h(\theta(t)) = \frac{1}{N}\sum_{k=1}^{N}(y_t - g(x_t,\theta(t)))g_\theta(x_t,\theta) - \frac{S_\beta}{N}\sum_{k=1}^{N}\Psi(x_t,\theta(t))\vartheta(t), \tag{19}$$

where $\vartheta = (\theta_1^2,\theta_2^2,\cdots,\theta_M^2)^T$ and $\Psi(x_t,\theta(t)) = g_{\theta\theta}(x_t,\theta(t))\mathbf{diag}\{g_\theta(x_t,\theta(t))\}$.

Clearly, the first term on the RHS of Equation (19) is proportional to the negative gradient of the MSE term. However, the anti-derivative of the second term is difficult. The corresponding objective function and the convergence proof can hardly be analyzed.

Except the case when the MLP output is linear, i.e. $g(x, w, v) = \sum_i w_i T_i(x, v)$, where w is the output weight vector and v is the input weight vector. $T_i(\cdot, \cdot)$ is the output of the i^{th} hidden unit. In such case, $\frac{\partial^2}{\partial w_i \partial w_j} g(x_t, w, v) = 0$. Therefore, enhancing fault tolerance of a MLP with linear output nodes cannot be achieved by simply adding noise to the output weights during training.

3.2 Injecting Additive Weight Noise

For the case that the injection weight noise is additive, the corresponding $h(\theta)$ can readily be obtained by replacing $A_\beta \theta(t)$ in Equation (16), Equation (17) and Equation (18) to β. Then,

$$h(\theta(t)) = \frac{1}{N} \sum_{k=1}^{N} (y_t - g(x_t, \theta(t))) g_\theta(x_t, \theta) - \frac{S_\beta}{N} \sum_{k=1}^{N} g_{\theta\theta}(x_t, \theta(t)) g_\theta(x_t, \theta(t)). \quad (20)$$

Clearly, the objective function minimized by $h(\theta(t))$ will be given by

$$\frac{1}{2N} \sum_{k=1}^{N} (y_t - g(x_t, \theta(t)))^2 + \frac{S_\beta}{2N} \sum_{k=1}^{N} \|g_\theta(x_t, \theta(t))\|^2. \quad (21)$$

Suppose the MLP is of linear output and additive weight noise is added only to the output layer, this objective function will become

$$\frac{1}{2N} \sum_{k=1}^{N} (y_t - g(x_t, \theta(t)))^2 + \frac{S_\beta}{2N} \sum_{k=1}^{N} \sum_i T_i^2(x, v). \quad (22)$$

The second term plays the role controlling the magnitude of the output of the hidden nodes.

Remark: One should note that the analysis in this section is purely heuristic, not analytically. Our analysis is focus on the expected updated equation, not the actual update equation. The reason is because the convergence proof for nonlinear system is not straight forward. As mentioned in [18], to prove the convergence of a nonlinear stochastic gradient descent algorithm, one needs to show that either (i) $\theta(t)$ can always be bounded or (ii) $\theta(t)$ can visit a local bound region infinite often. The two conditions are not easy to prove. Although, simulation results can also show that $\theta(t)$ is bounded for all t. Analytical proof has yet to be shown.

4 Conclusions

In this paper, analysis on the behavior of weight-noise-injection training has been presented. In contrast to the approach taken by An [1], we focus on the actual on-line update equation. From this, the convergence of weight-noise-injection training applying to RBF is proved analytically and the true objective function being minimized is revealed. Either for adding multiplicative or additive weight noise, it is found that the objective function being minimized is actually the mean square errors. Therefore, adding weight noise during training a RBF network can neither improve fault tolerance nor generalization.

For MLP, due to its nonlinearity, boundedness on $\theta(t)$ has yet been proven. Therefore, only analysis on the properties of the expected update equations has been presented. For the case of adding additive weight noise during training, it is shown that the objective function consists of two terms. The first term is the usual mean square term. But the second plays a role to regularize the magnitude of the output of the hidden units.

Acknowledgement. The research work reported in this paper is supported in part by Taiwan NSC Research Grant 97-2221-E-005-050.

References

1. An, G.: The effects of adding noise during backpropagation training on a generalization performance. Neural Computation 8, 643–674 (1996)
2. Bernier, J.L., et al.: Obtaining fault tolerance multilayer perceptrons using an explicit regularization. Neural Processing Letters 12, 107–113 (2000)
3. Bernier, J.L., et al.: A quantitative study of fault tolerance, noise immunity and generalization ability of MLPs. Neural Computation 12, 2941–2964 (2000)
4. Bishop, C.M.: Training with noise is equivalent to Tikhnov regularization. Neural Computation 7, 108–116 (1995)
5. Bolt, G.: Fault tolerant in multi-layer Perceptrons. PhD Thesis, University of York, UK (1992)
6. Bottou, L.: Stochastic gradient learning in neural networks. In: NEURO NIMES 1991, pp. 687–706 (1991)
7. Cavalieri, S., Mirabella, O.: A novel learning algorithm which improves the partial fault tolerance of multilayer NNs. Neural Networks 12, 91–106 (1999)
8. Chandra, P., Singh, Y.: Fault tolerance of feedforward artificial neural networks – A framework of study. In: Proceedings of IJCNN 2003, vol. 1, pp. 489–494 (2003)
9. Chiu, C.T., et al.: Modifying training algorithms for improved fault tolerance. In: ICNN 1994, vol. I, pp. 333–338 (1994)
10. Deodhare, D., Vidyasagar, M., Sathiya Keerthi, S.: Synthesis of fault-tolerant feedforward neural networks using minimax optimization. IEEE Transactions on Neural Networks 9(5), 891–900 (1998)
11. Edwards, P.J., Murray, A.F.: Fault tolerant via weight noise in analog VLSI implementations of MLP's – A case study with EPSILON. IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 45(9), 1255–1262 (1998)

12. Gladyshev, E.: On stochastic approximation. Theory of Probability and its Applications 10, 275–278 (1965)
13. Grandvalet, Y., Canu, S.: A comment on noise injection into inputs in backpropagation learning. IEEE Transactions on Systems, Man, and Cybernetics 25(4), 678–681 (1995)
14. Grandvalet, Y., Canu, S., Boucheron, S.: Noise injection : Theoretical prospects. Neural Computation 9(5), 1093–1108 (1997)
15. Hammadi, N.C., Hideo, I.: A learning algorithm for fault tolerant feedforward neural networks. IEICE Transactions on Information & Systems E80-D(1) (1997)
16. Jim, K.C., Giles, C.L., Horne, B.G.: An analysis of noise in recurrent neural networks: Convergence and generalization. IEEE Transactions on Neural Networks 7, 1424–1438 (1996)
17. Kamiura, N., et al.: On a weight limit approach for enhancing fault tolerance of feedforward neural networks. IEICE Transactions on Information & Systems E83-D(11) (2000)
18. Lai, T.L.: Stochastic approximation. Annals of Statistics 31(2), 391–406 (2003)
19. Leung, C.S., Sum, J.: A fault tolerant regularizer for RBF networks. IEEE Transactions on Neural Networks 19(3), 493–507 (2008)
20. Murray, A.F., Edwards, P.J.: Synaptic weight noise during multilayer perceptron training: fault tolerance and training improvements. IEEE Transactions on Neural Networks 4(4), 722–725 (1993)
21. Murray, A.F., Edwards, P.J.: Enhanced MLP performance and fault tolerance resulting from synaptic weight noise during training. IEEE Transactions on Neural Networks 5(5), 792–802 (1994)
22. Neti, C., Schneider, M.H., Young, E.D.: Maximally fault tolerance neural networks. IEEE Transactions on Neural Networks 3(1), 14–23 (1992)
23. Phatak, D.S., Koren, I.: Complete and partial fault tolerance of feedforward neural nets. IEEE Transactions on Neural Networks 6, 446–456 (1995)
24. Reed, R., Marks II, R.J., Oh, S.: Similarities of error regularization, sigmoid gain scaling, target smoothing, and training with jitter. IEEE Transactions on Neural Networks 6(3), 529–538 (1995)
25. Sequin, C.H., Clay, R.D.: Fault tolerance in feedforward artificial neural networks. Neural Networks 4, 111–141 (1991)
26. Sum, J., Leung, C.S., Hsu, L.: Fault tolerant learning using Kullback-Leibler Divergence. In: Proc. TENCON 2007, Taipei (2007)
27. Sum, J., Leung, C.S., Ho, K.: On objective function, regularizer and prediction error of a learning algorithm for dealing with multiplicative weight noise. IEEE Transactions on Neural Networks 20(1) (January 2009)
28. Sum, J., Leung, C.S., Ho, K.: On node-fault-injection training an RBF network. In: Designing Privacy Enhancing Technologies. LNCS. Springer, Heidelberg (2009)
29. Tchernev, E.B., Mulvaney, R.G., Phatak, D.S.: Investigating the Fault Tolerance of Neural Networks. Neural Computation 17, 1646–1664 (2005)

Intelligent Control of Heating, Ventilating and Air Conditioning Systems

Patrick Low Tiong Kie[1] and Lau Bee Theng[2]

[1] Kuching, Sarawak, Malaysia
[2] Swinburne University of Technology Sarawak Campus
School of Computing and Design, Sarawak, Malaysia
blau@swinburne.edu.my

Abstract. This paper proposed a simulation-optimization energy saving strategy for heating, ventilating and air conditioning (HVAC) systems' condenser water loop through intelligent control of single speed cooling towers' components. An analysis of system components has showed the interactions of control variables inside the cooling towers and between the cooling tower and chillers. Based on the analysis, a model based optimization approach was developed with evolutionary computation. A simulation application demonstrated the effectiveness of the proposed strategy. This strategy can also be easily modified and applied to single speed tools in the refrigerant loops.

1 Introduction

In modern HVAC systems, building cooling and heating sources consume the highest volume of the electricity. Consequently, the optimization of the building cooling and heating sources has been extensively studied in HVAC systems for more efficient part load operation using Variable Speed / Frequency drive (VSD/VFD). Global optimal control methods for plant cooling have been carried out by Sud [13], Lau et al. [9] and Johnson [8]. However those models are not for real time control. Braun et al. [2, 3, 4] has developed a strategy for optimal control of chilled water systems which was suitable for real time application. Yao et al. [15] has developed a mathematical model for optimizing cooling systems' operation based on energy analysis of the main dynamic facilities. On the other hand, Lu et al. [10] developed a global optimization strategy for heating, ventilating and air conditioning systems. However, the models do not consider the time dependent characteristics of parameters. Chang [5] presented an approach using Lagrangian method to solve the optimizing chiller loading. Chow et al. [6] introduced a concept of integrating artificial neural network and genetic algorithm optimization of absorption chillers. Fong et al. [7] introduced the evolutionary programming for optimizing chillers in the HVAC systems. Alcala et al. [1] developed a weighted linguistic fuzzy rules combined with a rule selection process for intelligent control of heating, ventilating and air conditioning systems concerning energy performance and indoor comfort requirements. Basically, most of the researches focus on chillers. The researches on HVAC systems discussed mainly on the newer systems in manufacturing plants that come with VSD to optimize the chiller water pumps, condenser water pumps, chillers and cooling towers [16]. The cost

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 927–934, 2009.

savings from VSD is very encouraging [14]. VFD is also used in dynamic Transient Simulation Program based simulation platform for alternative control strategies including set-point control logics of the supply cooling water temperature and cooling tower fan modulation control methods as well as different number control means of cooling towers [4]. However, upgrading the older HVAC systems which use single speed controllers requires huge costs (approx. USD30K for complete installation of single VSD/VFD) depending on the number and type of drives. Hence our proposed approach focuses on energy savings on these single speed HVAC systems through the computational intelligence. Energy saving on HVAC through computational intelligence optimization of cooling towers is feasible [10, 12]. Our main concern in modeling minimal power consumption for the whole condenser water loop is the single speed cooling tower fans and condenser water pumps.

2 Our Optimization Target: Condenser Water Loop

The main purpose of cooling towers is to supply condenser water to chillers using a condenser water loop. This is because the performance of a chiller is influenced by chilled water supply temperature (T_{shws}), condenser water supply temperature (T_{cws}) and cooling load (C). For cooling towers, the power consumptions of pumps and fans are influenced by two parameters. mass flow rates of water and the pressure difference between the inlets and outlets. The characteristics of pumps and fans are very similar. The plant's HVAC systems are simplified as shown in Fig. 1 with labels for components and measured readings. The main components are six cooling towers fans and six condenser water pumps servicing the whole chiller plant. The adjustment of single speed cooling tower fans and condenser water pumps has effects on the total water side heat load of cooling towers, H. Consequently, the heat load is affected by the mass flow and heat of water, condenser water temperature at inlet and outlet. Even though we aim to minimize the power consumption of cooling towers, the manipulation of the fans and pumps must be optimized to meet the cooling load of the chillers in the plant, C. The cooling load is measured from mass flow of chilled water, heat of water, chilled water supply and return temperatures. The chillers have two types of capacity, namely low temp and high temp. There are three low temp chillers which have a rated power of 1000 kW each and three high temp chillers which have a rated power of 750 kW each. The chillers and their chilled water pumps are controlled by variable speed drives. The total power consumption for chiller side is 5700 kW per hour. On the other hand, the estimation of cooling load (Equation 1), C [12] of the chillers is 3415.5 tons when they are fully operated. m_{chw} is the mass flow of chilled water, c_p is the heat of water under constant pressure, h_p is pounds of heat per galloon of water, T_{chwr} and T_{chws} are the chilled water return and supply temperatures. The chillers are set to perform at 80% of its designated capacity presently. Presently, the coefficient of performance, COP [12] of the low temp chiller is 2.00 and high temp chiller is 3.20 calculated using Equation 2. E_c represents energy contributed in Btu/h and P_a is the power required in watts. However, the total power consumption for the six cooling towers which consist of six single speed fans operating together with six single speed condenser water pumps are 715.2 kW per hour. Due to the mass flow of water at inlet ($m_{w,i}$) and outlet ($m_{w,o}$) has the same volume and makeup water (m_m) is

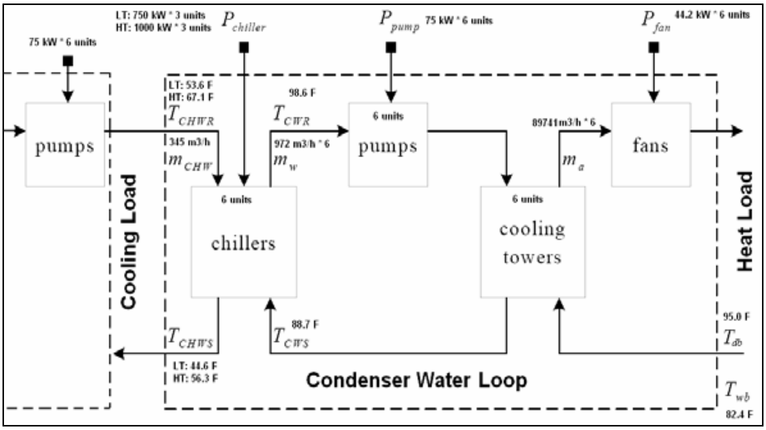


Fig. 1. Schematic view of condenser water loop

only 0.001% of the mass flow, Braun [2] model (ε_a) in Equation 3 is simplified. The heat transfer effectiveness, μ based on Braun's model is summarized in Equation 4. It utilizes the mass flow and temperature of water at both inlet and outlet. Ambient temperature is measured by wet bulb at inlet i.e. 82.4^{O}F. Hence the designated efficiency of the cooling tower based on Braun's heat transfer effectiveness, μ is 0.6111. The cooling towers have the designated heat load capacity; H [11] is 8468.064 tons when they are fully operated. m_w is the mass flow of condenser water, T_{cwr} and T_{cws} are the condenser water return and supply temperatures is shown in Equation 5.

$$C = \frac{c_p h_p m_{chw}(T_{chwr} - T_{chws})}{12000} \tag{1}$$

$$COP = (E_c / P_a)/3.412 \tag{2}$$

$$\varepsilon_a = \frac{m_{w,i} C_p T_{cwr} + m_m C_p T_m - m_{w,o} C_p T_{cws}}{m_{w,o} C_p (T_{cwr} - T_{wb})} \tag{3}$$

$$\mu = (T_{cwr} - T_{cws})/(T_{cwr} - T_{wb}) \tag{4}$$

$$H = \frac{c_p h_p m_w (T_{cwr} - T_{cws})}{15000} \tag{5}$$

3 Objective Function

Looking into the costs of investing in another dozen of VSD, and also the heat load, we probe into the possibility of modeling the single speed fans and condenser water pumps to meet the heat load and cooling load on real time basis. Experiments have been conducted to find out the possible hazards of manipulating the single speed fans and pumps. Hence it is feasible as it does not trip or interrupt other equipments in the plant. The objective function is to minimize total power consumption of the condenser water loop where P_{CT} is the measured total power consumption of cooling tower consists of condenser water pumps and cooling tower fans, P_{CH} is the measured total power consumption of chillers and chilled water pumps in Equation 6. The chillers have variable speed drives that control the individual cooling load based on the capacity, adjustment factor for part load and temperature as shown in the equation below. Hence the total power consumption of the chillers can be obtained directly from the controller. Due to the use of VSD, the chiller efficiency has been maximized based on various condenser water supply, chilled water supply and return temperatures. P_{CH} [11] is measured by $Q_{cap,i}$, nominal capacity, PLR_i , part load factor and T_i , temperature factor of chiller i as in Equation 7. The modeling of cooling tower power consumption, P_{CT} is determined by total power consumption by single speed pumps and fans, P_{pump} and P_{fan} as in Equation 8. The power consumption of condenser water pumps are calculated based on the rated power, measured and nominal mass flow of condenser water flow. Since it is a single speed pump, hence the power consumption is zero when it is turned off. a_p is the pump mode, pm_0 is the rated power, $m_{w,p}$ measures the mass flow of water and $m_{wn,p}$ measures the nominal mass flow of water of *i-th* pump as calculated in Equation 9. As for the power consumption of single speed cooling tower fan, the nominal and measured mass flow of air and rated power are calculated (Equation 10). Hence, if the fan is turned off, the power consumption is zero indicated by fan mode, b_p=0. fn_0 is the rated power, $m_{a,p}$ measures the mass flow of air and $m_{an,p}$ measures the nominal mass flow of air of *i-th* pump.

$$P_{\min} = P_{CT} + P_{CH} \tag{6}$$

$$P_{CH} = \sum_{i=1}^{i \le 6} Q_{cap,i} COP \cdot PLR_i \cdot T_i \tag{7}$$

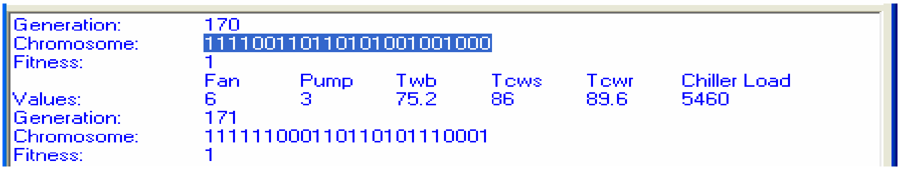
$$P_{CT} = P_{pump} + P_{fan} \tag{8}$$

$$P_{pump} = \sum_{p=1}^{p \le 6} a_p \frac{m_{w,p}}{m_{wn,p}} pm_o \tag{9}$$

$$P_{fan} = \sum_{f=1}^{f \le 6} b_p \frac{m_{a,p}}{m_{an,p}} fn_o \tag{10}$$

4 String Encoding

In order to meet our objective function, we model a set of related variables for optimization into a string which includes $(a_p, b_p, T_{wb}, T_{cws}, T_{cwr}, C)$ in Fig. 2. An initial population of random bit strings is obtained from the measured field data. The field data of this initial population is evaluated for their fitness or goodness in solving the problem. The initial population is evaluated to minimize P_{min} over the range of minimum and maximum values discussed. As P_{min} is nonnegative over the range, so it is used as the fitness of the string encoding.



```
Generation:     170
Chromosome:     111100110110101001001000
Fitness:        1
                Fan        Pump       Twb        Tcws       Tcwr       Chiller Load
Values:         6          3          75.2       86         89.6       5460
Generation:     171
Chromosome:     111111000110110101110001
Fitness:        1
```

Fig. 2. Strings and chromosome

5 Fitness Function

The fitness of a chromosome is evaluated based on the setting and fulfillment of the constraints. The fitness function, *f* is expressed in the following equation with penalties Pe_1 and Pe_2. Pe_1 assesses the chillers' cooling load, *C*. Pe_2 assesses the actual heat rejection capacity of cooling towers under the measured wet bulb, condenser water return and supply temperatures. The higher the fitness, *f* would signal the better the generation is. There are a few constraints that must be in place to validate the fitness function. All the variables must fall within the minimum and maximum allowed values.

$$f = \frac{1}{P_{\min} - (Pe_2 - Pe_1)} \tag{11}$$

$$Pe_1 = C \text{ And } Pe_2 = H \cdot (\frac{T_{cwr} - T_{cws}}{T_{cwr} - T_{wb}}) \tag{12}$$

$$H < C \text{ Where } H_{\min} \le H \le H_{\max} \text{ and } C_{\min} \le C \le C_{\max} \tag{13}$$

$$T_{cwr,\min} \le T_{cwr} \le T_{cwr,\max} \; T_{cws,\min} \le T_{cws} \le T_{cws,\max}\text{;} \; T_{cws,\min} \le T_{cws} \le T_{cws,\max}\text{;} \tag{14}$$

$$m_{w,\min} \le m_w \le m_{w,\max} \text{ and } m_{a,\min} \le m_a \le m_{a,\max} \tag{15}$$

Constraint 1: The heat load is measured in ton which has a minimum of 1800 ton to a maximum of 10800 from the six cooling towers. On the hand, the cooling load of six chillers measured in tons has a minimum of 569.25 tons to a maximum of 3757.05 tons. At any point of time, the heat load capacity must be larger than the cooling load performed by the chillers to prevent tripping.

Constraint 2: The temperatures of condenser water (supply and return) and wet bulb fall within the range of minimum and maximum values in Fahrenheit. Presently, the designated temperature for web bulb at inlet, T_{wb} is 82.4^0F, condenser water supply, T_{cws} is 88.7^0F and condenser water return, T_{cwr} is 98.6^0F.

Constraint 3: The cooling towers' mass flow of condenser water and air are modeled in m^3/h as m_w and m_a. They fall within the minimum and maximum designated capacity for optimized set points. Maximum air flow is achieved when all fan mode is 'on' that is 538446 m^3/h and water flow achieves its maxima when all pump mode is 'on' that is 5832 m^3/h.

After the initial population of 2160 chromosomes consists of the variables are evaluated for fitness, new population is generated using three genetic operators that are reproduction, crossover, and mutation. Each gene mutates with a probability of 0.1 and the crossover rate is 0.7. Once the operation setting of the single speed cooling tower fans and condenser water pumps have been optimized through the genetic operations, they are compared with the existing operating setting before been put in force. This is a safety measure to prevent the uncertainties of the genetic algorithm due to insufficient evolution time. If such a condition occurs, the system will operate at present set points without any changes until the next sampling period. In our project, the termination of genetic operations occur when there is NO better fitness value (the most minimum power consumption achieved) obtained from a generation. From

Table 1. GA parameters setting

Initial Population	2160	**Crossover**	0.7
Mutation	0.1	**No of generation**	200

our testing, we found out that a 200 generation would be when the fitness achieves its maxima.

6 Experimental Results

The simulation-optimization application focused on optimizing the control of the single speed cooling towers' fans and condenser water pumps on real time basis while the VSD controlled chillers are not interfered. The control affects the mass flow of air and water at both inlet and outlet of cooling towers which change the heat transfer effectiveness of cooling towers. The experiment aims to observe the optimized setting of cooling tower fans and pumps throughout 24 hours of a day based on ambient temperature, chillers' cooling load and cooling tower's heat transfer effectiveness. The average measured cooling tower efficiency based on Braun's model that takes into account the average ambient temperature at different hours of the day is shown in Fig.3. All the data measured were displayed in a daily average because the tropical climate here does not have distinctive seasonal difference. The averaged data is representative for the whole year. However, the experiment has been simulated for daily data of three months. The ambient temperatures vary between 68.9^0F to 82.4^0F throughout 24 hours of a day. Due to the ambient temperature achieves its maxima from 13:00 to 17:00; the heat transfer effectiveness of the cooling towers is the highest at these hours. On the other hand, due to the efficiency of cooling tower vary throughout 24 hours; the chillers' cooling load also varies accordingly where it achieves its maximum at 17:00. The ambient temperatures that change throughout the day have affected the chiller cooling load and cooling tower efficiency. This is due to higher ambient temperature at cooling tower inlet during day time has increased the condenser water supply temperature as well and vice versa during night time. We measure the wet bulb temperature at cooling tower inlet hourly throughout the day, and then measure the mass flow of air, mass flow of water, condenser water supply and return temperatures in the cooling towers. We optimized the mass flow and temperatures based on the fitness value of the population through the operating state of the fans and pumps. The mass flows are optimized with the constraint that cooling load of chillers can be met by heat transfer capacity of the cooling towers at any hour of the day. When the ambient temperature is lower, the calculated efficiency of the cooling tower is lower assuming the condition that condenser water supply and return temperatures remain constant. However, the condenser water supply temperature also decreases when outdoor temperature decreases at any time of the day. With these temperatures drop, we optimize the mass flow of air and water of the cooling towers. With the optimization by evolutionary computation through genetic algorithm, we manage to optimize the operation state of six pumps and six fans within the condenser water loop. Fig.3 also shows the hourly optimized power consumption of cooling towers (1st bar), chillers (2nd bar), optimized total power consumption (3rd bar), total power consumption before optimization (4th bar). The daily power consumption of these pumps and fans when fully operated is 17164.80kW daily which costs

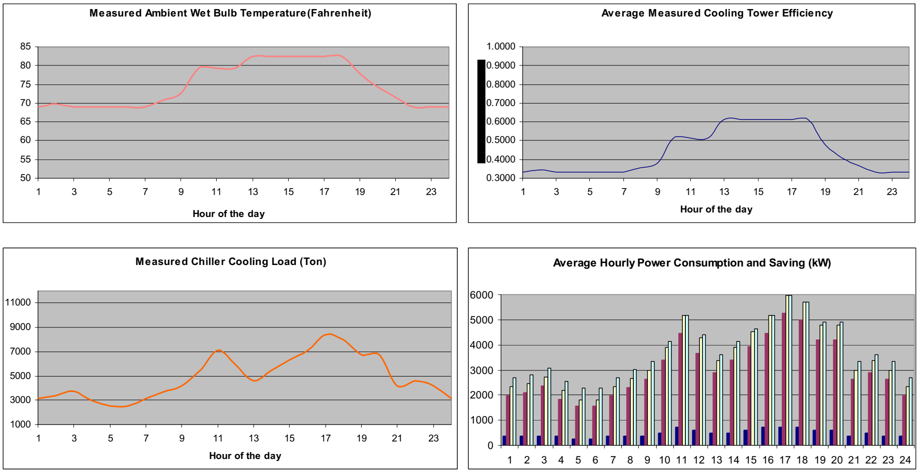


Fig. 3. Hourly Ambient Temperature, Chiller Cooling Load, Cooling Tower Efficiency and Power Consumption

USD1716.48 (approx. USD 0.10 per kW presently) daily and USD626515.20 annually. With the optimization of these components, we are able to cut down the energy consumption by 5,960 kW daily and 2,175,400 kW annually. This will give a cost saving of USD217540 annually. This gives a 34.7% saving on the total cooling tower power consumption or 7% of total condenser water loop power consumption as compare to fixed operation approach used previously.

7 Conclusions

This paper discussed a cost saving strategy through cooling towers in the condenser water loop. This strategy focuses on single speed control components in the loop where chillers and chilled water pumps which have variable speed drives are not modeled. The optimization cooling towers' water and air flows with fans and condenser water pumps through computational intelligence managed to cut down the total electricity consumed by cooling towers by 34.7% that is equivalent to USD 217,540 annually. The field data collection needs to be carried out for a longer term to observe any unusual behaviors of the parameters within the condenser water loop. With the more field data, optimization can be enhanced.

References

[1] Alcala, R., Casillas, J., Cordon, O., Gonzalez, A., Herrera, F.: A genetic rule weighting and selection process for fuzzy control of heating, ventilating and air conditioning systems. Engineering Applications of Artificial Intelligence 18, 279–296 (2005)

[2] Braun, J.E.: Methodologies for the design and control of chilled water systems. Ph.D. Thesis. University of Wisconsin (1988)

[3] Braun, J.E., Klein, S.A., Beckman, W.A., Mitchell, J.W.: Application of optimal control to chilled water systems without storage. ASHRAE Transactions 95(1), 663–675 (1989)

[4] Braun, J.E., Klein, S.A., Beckman, W.A., Mitchell, J.W.: Methodologies for optimal control of chilled water systems without storage. ASHRAE Transactions 95(1), 652–662 (1989)

[5] Chang, Y.C.: A novel energy conservation method-optimal chiller loading. Electric Power Systems Research 69(3), 221–226 (2004)

[6] Chow, T.T., Zhang, G.Q., Lin, Z., Song, C.L.: Global optimization of absorption chiller system by genetic algorithm and neural network. Energy and Buildings 34(1), 103–109 (2000)

[7] Fong, K.F., Hanby, V.I., Chow, T.T.: HVAC system optimization for energy management by evolutionary programming. Energy and Buildings 38, 220–231 (2006)

[8] Johnson, G.A.: Optimization techniques for a centrifugal chiller plant using a programmable controller. ASHRAE Transactions 91(2), 835–847 (1985)

[9] Lau, A.S., Beckman, W.A., Mitchell, J.W.: Development of computer control routines for a large chilled water plant. ASHRAE Transactions 91(1), 780–791 (1985)

[10] Lu, L., Cai, W., Chai, Y.S., Xie, L.: Global optimization for overall HVAC systems. Part I. Problem formulation and analysis. Energy Conversion and Management 46(7), 999–1014 (2005)

[11] MQuiston, F.C., Parker, J.D., Spitler, J.D.: Heating, ventilating and air conditioning: Analysis and design, 6th edn. John Wiley, Chichester (2005)

[12] Nabil, N., Samir, M., Mohammed, Z.: Self-tuning dynamic models of HVAC system components. Energy and Buildings 40, 1709–1720 (2008)

[13] Sud: Control strategies for minimum energy usage. ASHRAE Transactions 90(2), 247–277 (1984)

[14] Wang, S., Xu, X.: Effects of alternative control strategies of water-evaporative cooling systems on energy efficiency and plume control: A case study. Building and Environment 43, 1973–1989 (2008)

[15] Yao, Y., Lian, Z., Hou, Z., Zhou, X.: Optimal operation of a large cooling system based on an empirical model. Applied Thermal Engineering 24(16), 2303–2321 (2004)

[16] Yu, F.W., Chan, K.T.: Optimization of water-cooled chiller system with load-based speed control. In: Applied Energy, vol. 85, pp. 931–950. Elsevier, Amsterdam (2008)

Bregman Divergences and Multi-dimensional Scaling

Pei Ling Lai[1] and Colin Fyfe[2]

[1] Southern Taiwan University,
Tainan, Taiwan
pei_ling_lai@hotmail.com
[2] University of the West of Scotland, UK
colin.fyfe@uws.ac.uk

Abstract. We discuss Bregman divergences and the very close relationship between a class of these divergences and the regular family of exponential distributions before applying them to various topology preserving dimension reducing algorithms. We apply these to multidimensional scaling (MDS) and show the effect of different Bregman divergences. In particular we derive a mapping similar to the Sammon mapping. We apply these methods to face identification.

1 Introduction

Visualizing high dimensional data is problematic since we are not equipped with senses appropriate for this task. Indeed, even the task of converting two dimensional representations on our retina to the three dimensional representations our brain makes of the world is an inverse problem which is impossible to solve with 100% accuracy. Therefore we search for low dimensional representations of high dimensional data which capture some intrinsically interesting properties of the data. One such property is capturing the local distances in the high dimensional data and trying to maintain these relationships in a low dimensional projection of the data [4].

Bregman divergences have recently received a great deal of interest recently in terms of clustering and finding unsupervised projections of a data set [2,6,5,3,1]. In this paper, we investigate three groups of algorithms which use Bregman divergences as the distance measures. The first is multidimensional scaling (MDS), a standard technique in the literature. We show that Bregman divergences give a simple bridge between classical MDS and the popular Sammon mapping.

2 Bregman Divergences

Consider a strictly convex function $F : S \rightarrow \Re$ defined on a convex set $S \subset \Re^d$. A Bregman divergence between two elements, p and q, of S is defined to be

$$d_F(p,q) = F(p) - F(q) - \langle (p-q), \nabla F(q) \rangle \tag{1}$$

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 935–942, 2009.

where the angled brackets indicate an inner product and $\nabla F(q)$ is the derivative of F evaluated at q. This can be viewed as the difference between $F(p)$ and its truncated Taylor series expansion around q. Thus it can be used to 'measure' the convexity of F: Figure 1 illustrates how the Bregman divergence is the difference between $F(p)$ and the value which would be reached from $F(q)$ with a linear change for $\nabla F(q)$.

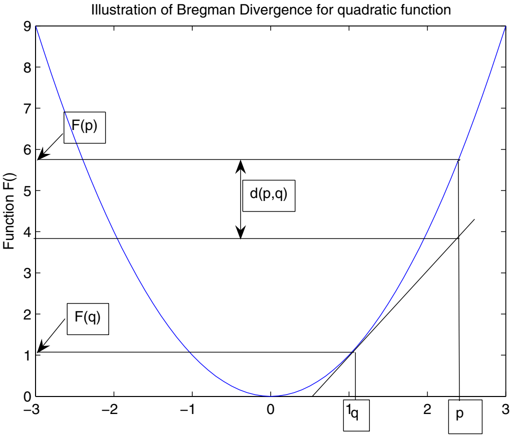


Fig. 1. The divergence is the difference between $F(p)$ and the value of $F(q) + (p - q)\nabla F(q)$

Example 1. The squared Euclidean distance is a special case of the Bregman divergence in which $F(.) = \| \, . \, \|^2$

$$\begin{aligned} d_F(\mathbf{x}, \mathbf{y}) &= \| \mathbf{x} \|^2 - \| \mathbf{y} \|^2 - \langle \mathbf{x} - \mathbf{y}, \nabla F(\mathbf{y}) \rangle \\ &= \| \mathbf{x} \|^2 - \| \mathbf{y} \|^2 - \langle \mathbf{x} - \mathbf{y}, 2\mathbf{y} \rangle \\ &= \| \mathbf{x} - \mathbf{y} \|^2 \end{aligned}$$

Example 2. The Kullback-Leibler divergence is another special case in which $F(\mathbf{p}) = \sum_{j=1}^{d} p_j \log p_j$. Consider two discrete probability distributions, $\mathbf{p}$ and $\mathbf{q}$.

$$\begin{aligned} d_F(\mathbf{p}, \mathbf{q}) &= \sum_{j=1}^{d} p_j \log_2 p_j - \sum_{j=1}^{d} q_j \log_2 q_j - \langle \mathbf{p} - \mathbf{q}, \nabla F(\mathbf{q}) \rangle \\ &= \sum_{j=1}^{d} p_j \log_2 p_j - \sum_{j=1}^{d} q_j \log_2 q_j \\ &\quad - \sum_{j=1}^{d} (p_j - q_j)(\log_2 q_j + \log_2 e) \end{aligned}$$

$$= \sum_{j=1}^{d} p_j \log_2 \frac{p_j}{q_j} - log_2 e \sum_{j=1}^{d} (p_j - q_j)$$

$$= \sum_{j=1}^{d} p_j \log_2 \frac{p_j}{q_j} = K.L.(\mathbf{p} \parallel \mathbf{q})$$

since $\sum_{j=1}^{d} p_j = \sum_{j=1}^{d} q_j = 1$. This divergence can be used with general vectors (i.e. not necessarily probability distributions) and then we get the Generalised I-divergence, $d_F(\mathbf{p}, \mathbf{q}) = \sum_{j=1}^{d} p_j \log \frac{p_j}{q_j} - \sum_{j=1}^{d} (p_j - q_j)$. Other divergences include the Itakura-Saito divergence, the Mahalanobis distance and the logistic loss, corresponding to $F(x) = \log(x), F(\mathbf{x}) = \mathbf{x}\Sigma^{-1}\mathbf{x}$, with Σ the data covariance matrix, and $F(x) = x \log x + (1-x)\log(1-x)$ respectively.

2.1 Properties of Bregman Divergences

First note that, in general, $d_F(\mathbf{p}, \mathbf{q}) \neq d_F(\mathbf{q}, \mathbf{p})$. However we can create symmetric divergences:

$$\begin{aligned} S_F(\mathbf{p}, \mathbf{q}) &= \frac{1}{2}(d_F(\mathbf{p}, \mathbf{q}) + d_F(\mathbf{q}, \mathbf{p})) \\ &= \frac{1}{2}\langle \mathbf{p} - \mathbf{q}, \nabla F(\mathbf{p}) - \nabla F(\mathbf{q}) \rangle \end{aligned}$$

This gives us a divergence measured on the space S and its derivative space ∇S.

All Bregman divergences satisfy

Non-negativity $d_F(\mathbf{p}, \mathbf{q}) \geq 0$ with equality if and only if $\mathbf{p} = \mathbf{q}$.
Convexity but only guaranteed in the first parameter.
Linearity $d_{aF_1+bF_2}(\mathbf{p}, \mathbf{q}) = a d_{F_1}(\mathbf{p}, \mathbf{q}) + b d_{F_2}(\mathbf{p}, \mathbf{q})$

A fuller description of the properties of Bregman divergences can be found in [2].

However, this leaves open the question as to which Bregman divergence is the best one to use for any particular data set, something which will inevitably depend on the distribution of the data set. To find an answer to this, we digress to re-state the properties of the exponential family of distributions.

3 The Exponential Family

[2] have shown that there is bijection between a set of Bregman divergences and members of the regular exponential family of probability distributions. The exponential family of distributions is a surprisingly wide family whose members have distributions of the form

$$p_{G,\theta}(\mathbf{x}) = \exp(\langle \mathbf{t}(\mathbf{x}), \theta \rangle - G(\theta)) p_0(\mathbf{t}(\mathbf{x})) \tag{2}$$

where $\mathbf{t}(\mathbf{x})$ is known as the natural statistic, θ is known as the natural parameter and $G(\theta)$ is the cumulant function which defines the exponential family. An example of the exponential family are

The 1 dimensional Gaussian with unit variance

$$\begin{aligned} p_{G,\theta}(x) &= \frac{1}{\sqrt{2\pi}\sigma} e^{(-\frac{(x-\mu)^2}{2})} \\ &= \frac{e^{-x^2}}{\sqrt{2\pi}} e^{x\mu - \frac{\mu^2}{2}} \\ \text{So that } t(x) &= x \\ \theta &= \mu \\ \text{and } G(\theta) &= \frac{\theta^2}{2} = \frac{\mu^2}{2} \end{aligned}$$

Other well known members of this family are the bernoulli, multinomial, beta, Dirichlet, Poisson, Laplace, gamma and Rayleigh distributions. In the remainder of this paper, we consider only regular exponential families in which $\mathbf{t}(\mathbf{x}) = \mathbf{x}$.

We define the expectation of X with respect to $p_{G,\theta}$ to be

$$\mu = E_{p_{G,\theta}}[X] = \int_{\Re^d} \mathbf{x} p_{G,\theta}(\mathbf{x}) d\mathbf{x} \tag{3}$$

It can be shown [2] that there is a bijection between the set of expected values, μ, and the set of natural parameters, θ. In fact, let d_F be the Bregman divergence corresponding to the distribution, $p_{G,\theta}$. Then let $g() = \nabla G$ and let $f = \nabla F$. Then $\mu = g(\theta)$ and $\theta = f(\mu)$, which is readily verified for the distributions above.

Consider a member of the regular exponential family with known cumulant function, $G(\theta)$. Then $G(.)$ is a closed convex function. Define its conjugate function as

$$F(\mathbf{x}) = \sup_{\theta} \{ \langle \mathbf{x}, \theta \rangle - G(\theta) \} \tag{4}$$

Then there is an unique θ^* which attains the supremum and $F()$ is also a convex function. If the domain of F is S and the domain of G is Θ, then (S, F) is the Legendre dual of (Θ, G). In particular, there exists a θ such that $F(\mu) = \langle \mu, \theta \rangle - G(\theta)$. Differentiating and setting the derivative to 0, we see that $g(\theta) = \mu$ and $f(\mu) = \theta$; then since $G()$ is strictly convex, $F()$ is too and so can be used to define a Bregman divergence. Consider two members of an exponential family with natural parameters, θ_1 and θ_2, and expectations, μ_1 and μ_2. Then it can be shown that minimising the Bregman divergence with respect to the cumulant function between the natural parameters is equivalent to minimising the Bregman divergence with respect to the dual function (but in the opposite direction) between the expectations. Also it can be shown that maximising the likelihood of a data set is equivalent to minimising the associated Bregman divergence between the mean of the distribution and the data.

In practical terms, we might fit a particular member of the exponential family to a data set which means we have determined the cumulant function, $G(.)$. We then identify the dual function, $F(.)$, based on which we can find the Bregman divergence $d_F(.)$ knowing that minimising the Bregman divergence between the mean of the distribution and its natural statistics maximises the log likelihood of the distribution under this probability density function.

4 Multi-dimensional Scaling

We illustrate the behaviour of the Bregman divergences with the Sammon mapping. Multidimensional scaling (MDS) finds a low dimensional (typically two dimensional) representation of each data point such that the distances in this representation which we will call latent space are as close to the original (Euclidean) distances as possible. Thus if we wish the latent points $\mathbf{x}_i$ to represent the data points $\mathbf{y}_i$, we wish the distance d_{ij} between latent points $\mathbf{x}_i$ and $\mathbf{x}_j$ to be as close as possible to the distance, δ_{ij}, between data points $\mathbf{y}_i$ and $\mathbf{y}_j$. Thus the performance function for classical metric MDS is

$$J_{MDS} = \sum_{i,j=1}^{N} (d_{ij} - \delta_{ij})^2 \tag{5}$$

There is a well-known close relationship between the $\mathbf{x}_i$ and Principal Component Analysis (PCA) in that the final values of the $\mathbf{x}_i$ are the projections of the $\mathbf{y}_i$ on the principal axes found by PCA. Thus there are some important statistical properties associated with classical MDS.

The Sammon mapping weights each difference in (5) by the inverse of the distances in data space so that presevation of small distances is more important than preservation of distances between points which are very far apart. Its performance function is given by

$$J_{Sammon} = C \sum_{i=1,i<j}^{N} \frac{(d_{ij} - \delta_{ij})^2}{\delta_{ij}} \tag{6}$$

where $C = \frac{1}{\sum_{i=1,i<j}^{N} \delta_{ij}}$ is a normalising constant. Both the distances in data space and distances in latent space are often Euclidean. This algorithm has proved very popular in practice since it emphasises the maintenance of local distance relations at the expense of loosening constraints which are determined by data points far from one another. We will show that a Bregmanised version of classical MDS (5) has properties very similar to the Sammon mapping (6). However first we will investigate Bregman divergences in data space.

4.1 Simulations

We create 100 samples of 105 dimensional data. Each data sample is a noisy version of binary data with 95 zeros and 10 ones so that $\mathbf{y}_i$ has a 1 at positions $i, i+1, i+2, i+3, i+4$ and at $j, j+1, j+2, j+3, j+4$ where the starting point j is randomly chosen from a permutation of 1,...,100. Thus the i^{th} data sample will have an overlap of at least 4 with the $(i+1)^{th}$ and $(i-1)^{th}$ samples (when they exist) and could have an overlap of up to 8. Thus a two dimensional representation should reveal this. One sample of the dataset is shown in Figure 2: the 31st sample has 1s at positions 31-35 and 98-102.

The KL and GI divergences are more informative than the Euclidean distance in this case though we do not wish to state that this will always be the case for all datasets.

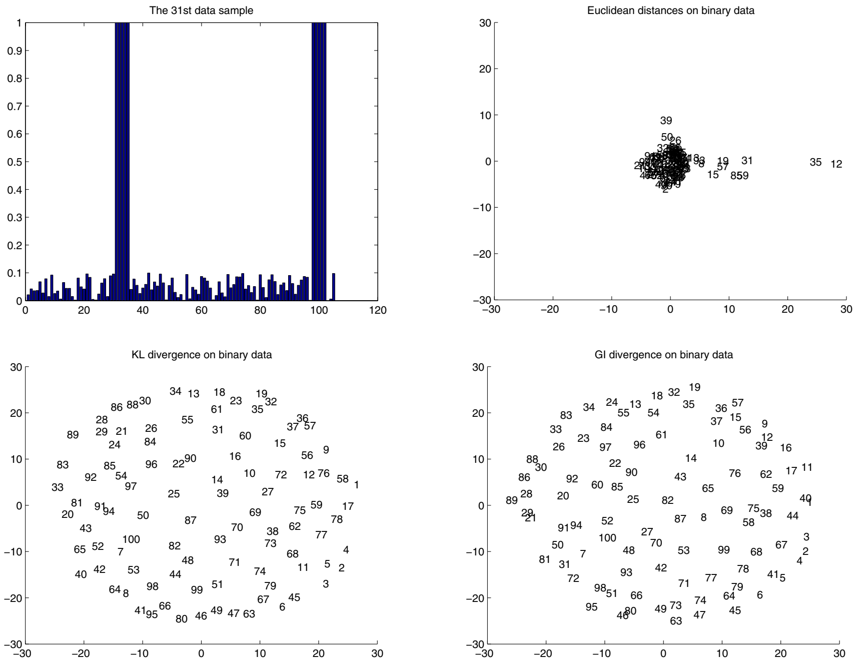


Fig. 2. Top left: one sample of the noisy binary data: it has two contiguous blocks each of 5 inputs which have value 1 while the remaining inputs each have small random values. The other 3 diagrams show Sammon mapping representations using different underlying Bregman divergences.

4.2 Alternative Mappings

In the previous section, we have illustrated Bregman divergences on data space but retained Euclidean distances in the latent space. However we can easily conceive of Bregman divergences in latent space. Thus we would be retaining the pseudo metric on the space while reducing the dimensionality of the space. Indeed we can conceive of a mapping which minimimises the Bregman divergence between the divergences between the two spaces. Thus a fully Bregmanised multidimensional scaling would minimise

$$J_{BMDS} = \sum_{i,j=1}^{N} d_{F_1}(d_{F_2}(\mathbf{x}_i, \mathbf{x}_j), d_{F_3}(\mathbf{y}_i, \mathbf{y}_j)) \tag{7}$$

For example, we may choose to have the Itakura-Saito divergence for F_1 i.e. $F_1() = -\log()$. Thus using, as before, d_{ij} for the divergence between latent points $\mathbf{x}_i$ and $\mathbf{x}_j$, i.e. $d_{ij} = d_{F_2}(\mathbf{x}_i, \mathbf{x}_j)$ and δ_{ij} for the divergence between data points $\mathbf{y}_i$ and $\mathbf{y}_j$ i.e. $\delta_{ij} = d_{F_3}(\mathbf{y}_i, \mathbf{y}_j)$, we have

$$d_F(d_{ij}, \delta_{ij}) = \log\left(\frac{\delta_{ij}}{d_{ij}}\right) + \frac{d_{ij} - \delta_{ij}}{\delta_{ij}} \tag{8}$$

We note the automatic scaling by δ_{ij} for MDS with Bregman divergences which the Sammon mapping heuristically inserts. δ_{ij} appears first inside a logarithm of the cost function and thus large values have less effect than they would otherwise and secondly it appears as a normalising term in the second part of the cost function and so makes differences between large distances in data space less important.

We can then create the update rule

$$\varDelta d_{ij} \propto -\frac{\partial d_F(d_{ij}, \delta_{ij})}{\partial d_{ij}} = \frac{1}{d_{ij}} - \frac{1}{\delta_{ij}} \tag{9}$$

If we retain an Euclidean distance in latent space, we may use the incremental update rule

$$\varDelta \mathbf{x}_i = \eta \sum_{j=1}^{N} (\mathbf{x}_i - \mathbf{x}_j)(\frac{1}{d_{ij}} - \frac{1}{\delta_{ij}}), \forall i \tag{10}$$

where η is a learning rate, and alternate this with a re-calculation of the distances between the points in latent space. We call this pseudo metric multidimensional Scaling. Note that this can be used regardless of which divergence is used in data space which needs only be calculated once. Thus (10) can be used whenever we have the Itakura-Saito divergence as F_1, the Euclidean distance as F_2 and any divergence for F_3.

5 Conclusion

We have reviewed Bregman divergences and shown the very close relationship between these divergences and the exponential family of probability density functions: if we know the pdf of a data set, we can choose the optimal divergence associated with that dataset.

In particular we have applied Bregman divergences to multidimensional scaling. We have shown multidimensional scaling mappings with Bregman divergences determining the 'distances' between points; we further showed that, starting with classical MDS but using the Itakura-Saito divergence between the distances, we create a mapping which is not unlike the Sammon mapping and has certainly interesting visualization properties.

References

1. Azoury, K.S., Warmouth, M.K.: Relative loss bounds for on-line density estimation with the exponential family of distributions. Machine Learning (43), 211–246 (2001)
2. Banerjee, A., Meruga, S., Dhillon, I., Ghosh, J.: Clustering with bregman divergences. Journal of Machine Learning Research 6, 1705–1749 (2005)

3. Collins, M., Dasgupta, S., Shapire, R.E.: A generalization of principal component analysis to the exponential family. In: NIPS 14 (2002)
4. Lee, J.A., Verleysen, M.: Nonlinear Dimensionality Reduction. Springer, Heidelberg (2007)
5. Neilsen, F., Boissonnat, J.-D., Nock, R.: Bregman voronoi diagrams: Properties, algorithms and applications (submitted, 2007)
6. Neilsen, F., Boissonnat, J.-D., Nock, R.: On bregman voronoi diagrams. In: ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 746–755 (2007)

Collective Activations to Generate Self-Organizing Maps

Ryotaro Kamimura

IT Education Center, Tokai University,
1117 Kitakaname Hiratsuka Kanagawa 259-1292, Japan
ryo@cc.u-tokai.ac.jp

Abstract. In this paper, we propose a new method called *collective activations* to realize self-organizing maps. We suppose that all neurons collectively respond to input stimuli, and this collectiveness is represented by the sum of all neurons' activations. Learning consists of imitating these collective activations as much as possible. We applied the method to artificial data and a broadband survey problem. In all these problems, we could obtain self-organizing maps similar or, in some cases, superior to those obtained by conventional SOM. Thus, the present study is considered to be the first step toward more realistic self-organizing maps.

1 Introduction

In this paper, we present a new method called *collective activations* to realize self-organizing maps without competitive learning by the winner-take-all algorithm. Competitive learning has been used as the base for many learning methods. The self-organizing map by Kohonen, especially, has attracted much attention, because it has given strong visualization performance. The conventional SOM is based upon the winner-take-all algorithm. However, it is not plausible that this winner-take-all mechanism is built in living systems. It seems to us that the softer mechanism of competition and cooperation should be developed.

Many attempts have been made to formulate more realistic procedures for competitive learning as well as self-organization. Among them, due attention has been paid to the information-theoretic approach for self-organizing maps [1], [2], [3], [4]. Linsker, especially, tried to formulate self-organizing maps by maximizing mutual information between input and output signals. These studies have shown that competitive learning is only one aspect of mutual information maximization [2], [3], [4]. However, in Linsker's methods, only complicated learning rules were formulated, and they have not been used in practical applications.

To overcome the shortcoming of Linsker's maximum information principle, we propose a method called *collection activations* to replace the common term "lateral interaction" and to stress that this collectiveness is one of the key elements in formulation. We suppose that all neurons collectively respond to input stimuli. This collectiveness is realized by summing all competitive unit activations. The summation is actually a weighted sum of all neurons. One of the main points for our model is that learning consists of imitating these collective activations. Usually, this collectiveness is realized by a Gaussian neighborhood function.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 943–950, 2009.

2 Theory and Computational Methods

2.1 Collective Activations

In this paper, we suppose that all neurons collectively respond to input stimuli. This phenomenon has been well known as lateral interaction among neurons. Lateral interactions or cooperation among neurons is adjacent or peripheral to competition among neurons. Our main point is that this lateral interaction should be imitated by competitive processes. As already mentioned, we used *collective activations* to stress the importance and the priority of lateral interaction.

Now, let us explain collective activations. Figure 1 shows a network architecture that is composed of a competitive and a collective layer. In the network, x_k^s denotes the kth element of the sth input pattern, w_{jk} represents connection weights from the kth input unit to the jth competitive unit and L is the number of input units. An output from an individual competitive unit can be computed by

$$v_j^s = \exp\left(- \frac{\sum_{k=1}^{L} (x_k^s - w_{jk})^2}{2\sigma^2} \right), \tag{1}$$

where σ denotes a Gaussian width. Figure 1 shows that the jth neuron cooperates with all the other neurons on the map and responds to input patterns. We realize this cooperation by summing all neighbors' unit activities. Then, we have collected activations

$$V_j^s = \sum_{m=1}^{M} W_{jm} v_m^s, \tag{2}$$

where W_{jm} denotes connection weights from the mth competitive unit to the jth competitive unit, and M is the number of competitive units. We can imagine many kinds of collectiveness on the map. We usually use the distance between competitive units for expressing collectiveness; for example, when competitive units are closer to their neighbors, they should be linked to them more intensely.

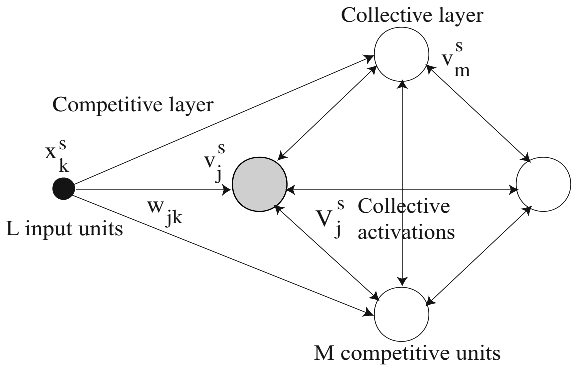


Fig. 1. Network architecture and how to compute collective activations

A distance function between two neurons can be defined by

$$W_{jm} = \exp\left(-\frac{(r_j - r_m)^2}{2\sigma^2}\right), \tag{3}$$

where r_j denotes a position of the jth unit and r_m denotes a position for the mth neighboring neuron. Thus, we have

$$V_j^s = \sum_{m=1}^{M} \exp\left(-\frac{(r_j - r_m)^2}{2\sigma^2}\right) v_m^s. \tag{4}$$

We can compute a normalized activity

$$q(j \mid s) = \frac{V_j^s}{\sum_{m=1}^{M} V_m^s}. \tag{5}$$

This normalized activity is considered to represent collective firing rates.

2.2 Imitating Collective Activations

We have defined collective activations, and now what we have to do is to obtain update rules to realize these collective activations. Suppose that $p(j|s)$ denotes a firing probability of the jth neuron. Therefore, we should make these probabilities as close as possible to the firing probabilities of collective neurons or firing rates. Thus, we have an objective cross entropy function defined by

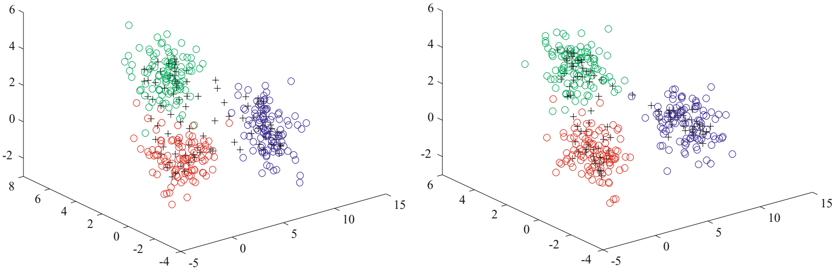
$$S = \sum_{s=1}^{S} p(s) \sum_{j=1}^{M} p(j \mid s) \log \frac{p(j \mid s)}{q(j \mid s)}. \tag{6}$$

It is possible to differentiate this cross entropy to obtain update rules, as Linsker did [3]. However, as is the case with Linsker's formulation [3], the rules become complicated update rules with heavy computation for computing conditional probabilities. Fortunately, we can skip complicated computation of conditional entropy by introducing the free energy used in statistical mechanics [5], [6], [7], [8], [9], [10], [11]. Following statistical mechanics, let us introduce free energy or a free energy-like function defined by

$$F = -2\sigma^2 \sum_{s=1}^{S} p(s) \log \sum_{j=1}^{M} q(j \mid s) \exp\left(-\frac{\sum_{k=1}^{L}(x_k^s - w_{jk})^2}{2\sigma^2}\right). \tag{7}$$

An optimal state specifies the actual output

$$p(j \mid s) = \frac{q(j \mid s) \exp\left(-\frac{\sum_{k=1}^{L}(x_k^s - w_{jk})^2}{2\sigma^2}\right)}{\sum_{j=1}^{M} q(j \mid s) \exp\left(-\frac{\sum_{k=1}^{L}(x_k^s - w_{jk})^2}{2\sigma^2}\right)}. \tag{8}$$



(a) SOM

(b) Collective activations

Fig. 2. Artificial data with final connection weights in black plus symbols by SOM (a) and collective activations (b)

Then, putting $p(j \mid s)$ into the cross entropy, we have

$$F = \sum_{s=1}^{S} p(s) \sum_{j=1}^{M} p(j \mid s) \sum_{k=1}^{L} (x_k^s - w_{jk})^2$$
$$+ 2\sigma^2 \sum_{s=1}^{S} p(s) \sum_{j=1}^{M} p(j \mid s) \log \frac{p(j \mid s)}{q(j \mid s)}. \tag{9}$$

Thus, to decease the free energy, we must decrease the cross entropy and the corresponding error function. By differentiating the free energy, we have update rules

$$\Delta w_{jk} = \beta \sum_{s=1}^{S} p(s) p(j \mid s)(x_k^s - w_{jk}), \tag{10}$$

where β is a learning rate and $p(s)$ is set to $1/S$ for all experiments discussed below.

2.3 Cooling Processes

One of the easiest ways to decrease the free energy is to decrease the Gaussian width. This is called a *cooling process*. We use a method from the SOM toolbox[1], and we decrease the parameter σ by

$$\sigma(t) = \sigma(0) \left(\frac{0.005}{\sigma(0)} \right)^{\frac{t}{T}} + \frac{1}{\sigma(0)}, \tag{11}$$

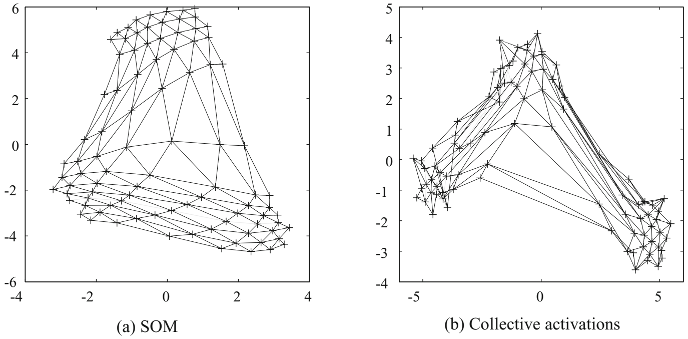
and the learning rate β is decreased by

$$\beta(t) = \sigma(0) \left(\frac{0.005}{\sigma(0)} \right)^{\frac{t}{T}}, \tag{12}$$

[1] http://www.cis.hut.fi/research/som-research.

where t is a learning epoch. The maximum number of epochs T is set to 1,000. The parameter $\sigma(0)$ is set to half of the map size. The term $1/\sigma(0)$ is introduced to stabilize learning.

3 Results and Discussion



(a) SOM

(b) Collective activations

Fig. 3. Sammon map obtained by SOM(a) and by collective activations (b)

3.1 Artificial Data

For this experiment, we used random vectors in three Gaussian kernels, the centers of which are at (0, 0, 0), (3, 3, 3) and (9, 0, 0)[2]. Figure 2 shows data with final connection weights in plus symbols produced by SOM(a) and by collective activations (b). As can be seen in Figure 2(a), final connection weights in plus symbols are fairly well located in three groups. However, several weights seem to be outside of the three groups. On the other hand, when we use the collection activations in Figure 2(b), final connection weights in plus symbols are completely located inside the three groups. Figure 3 shows Sammon maps obtained by SOM (a) and by collective activations (b). With SOM, the Sammon map is widely distributed. On the other hand, by using the collective activations in Figure 3(b), three groups are clearly separated. Finally, Figure 4 shows U-matrices and component planes produced by SOM (a) and by collective activations (b). Comparing the U-matrix produced by collective activations with those produced by SOM, we can say that with the use of collective activations, boundaries are clearer, meaning that the U-matrix obtained by the collective activations can divide input patterns into three groups more clearly. When we compare component planes obtained by collective activations and by SOM, we can also clearly see that the collective activations more explicitly detect the characteristics of input patterns.

[2] http://www.cis.hut.fi/research/som-research.

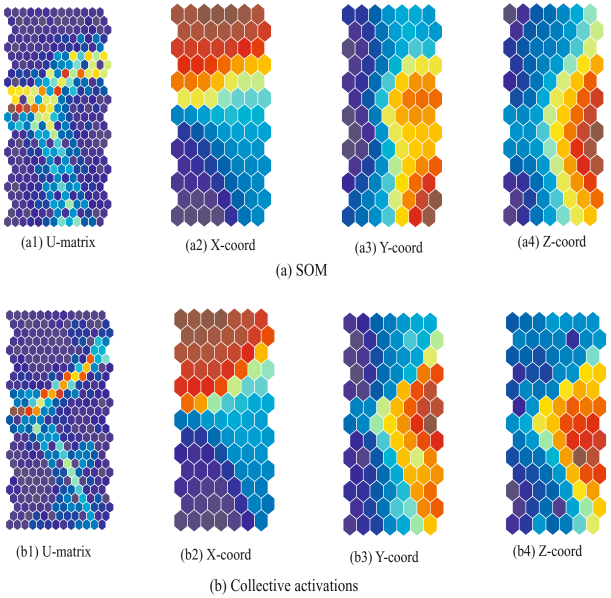


Fig. 4. U-matrices and component planes obtained by SOM (a) and collective activations (b). Warmer and cooler colors represent higher and lower values, respectively.

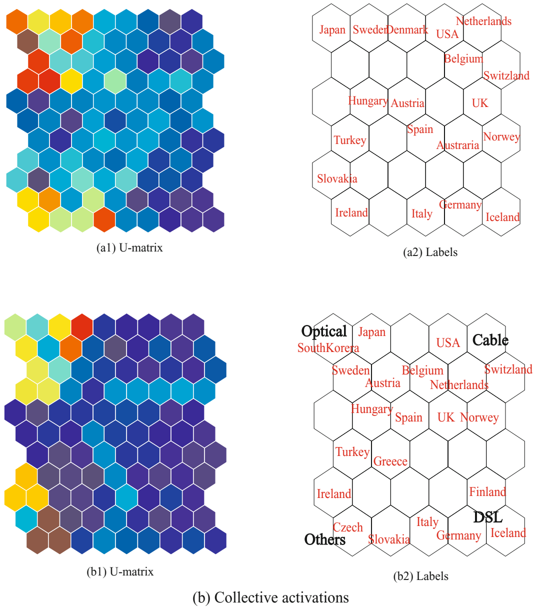


Fig. 5. U-matrix (1) and labels (2) obtained by SOM (a) and by collective activations (b)

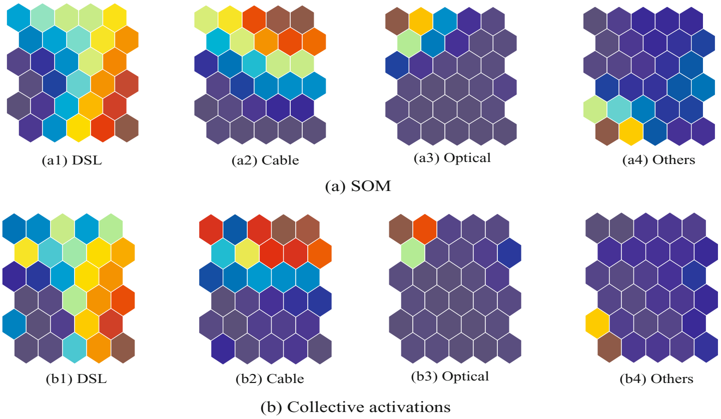


Fig. 6. Component planes obtained by conventional SOM (a) and collective activations (b)

3.2 Broadband Statistics

In this section, we examine the number of broadband access methods in OECD countries[3]. Figure 5(a) shows a U-matrix and labels obtained by conventional SOM. We cannot see any explicit boundaries in the U-matrix. On the other hand, by using collective activations, as shown in Figure 5(b), clearer boundaries, though their intensity is low, in light blue, can be detected. By examining component planes in Figure 6(a) and (b), we can see that, by those boundaries, countries are classified into four groups based upon the corresponding four access methods, that is, *optical*, *cable*, *DSL* and others.

4 Conclusion

We suppose that all neurons collectively respond to input stimuli. This collectiveness is realized by summing all neurons' activities. We train networks to imitate these collective activations in learning. We have proposed a new method called *collective activations* to realize self-organizing maps. When we applied the method to artificial data and a broadband problem, in all three cases, we obtained self-organizing maps comparable or superior to those obtained by conventional SOM. One of the major problems is how to cool learning processes. Depending upon different cooling processes, different final representations could be obtained. Thus, we should more exactly examine relations between cooling processes and final representations. Finally, some have already noticed close relations between our method (Linsker-like) and self-organizing mixture networks [12]. Thus, we should mathematically examine those relations for future development of this study.

[3] http://www.e-stat.go.jp/SG1/estat/eStatTopPortal.do.

Acknowledgment

The author is very grateful to Mitali Das for her valuable comments.

References

1. Hulle, M.M.V.: The formation of topographic maps that maximize the average mutual information of the output responses to noiseless input signals. Neural Computation 9(3), 595–606 (1997)
2. Linsker, R.: Self-organization in a perceptual network. Computer 21, 105–117 (1988)
3. Linsker, R.: How to generate ordered maps by maximizing the mutual information between input and output. Neural Computation 1, 402–411 (1989)
4. Linsker, R.: Local synaptic rules suffice to maximize mutual information in a linear network. Neural Computation 4, 691–702 (1992)
5. Ueda, N., Nakano, R.: Deterministic annealing variant of the em algorithm. In: Advances in Neural Information Processing Systems, pp. 545–552 (1995)
6. Rose, K., Gurewitz, E., Fox, G.C.: Statistical mechanics and phase transition in clustering. Physical review letters 65(8), 945–948 (1990)
7. Martinez, T.M., Berkovich, S.G., Schulten, K.J.: Neural-gas network for vector quanitization and its application to time-series prediction. IEEE transactions on neural networks 4(4), 558–569 (1993)
8. Erdogmus, D., Principe, J.: Lower and upper bounds for misclassification probability based on renyi's information. Journal of VLSI signal processing systems 37(2/3), 305–317 (2004)
9. Torkkola, K.: Feature extraction by non-parametric mutual information maximization. Journal of Machine Learning Research 3, 1415–1438 (2003)
10. Kamimura, R.: Free energy-based competitive learning for mutual information maximization. In: Proceedings of IEEE Conference on Systems, Man, and Cybernetics, pp. 223–227 (2008)
11. Kamimura, R.: Free energy-based competitive learning for self-organizing maps. In: Proceedings of Artificial Intelligence and Applications, pp. 414–419 (2008)
12. Yin, H., Allinson, N.M.: Self-organizing mixture networks for probability density estimation. IEEE Transactions on Neural Networks 12(2), 405–411 (2001)

A Closed-Form Estimator of Fully Visible Boltzmann Machines

Jun-ichiro Hirayama and Shin Ishii

Graduate School of Informatics, Kyoto University
Gokasho, Uji, Kyoto 611-0011, Japan
{junich-h@sys.,ishii@}i.kyoto-u.ac.jp
http://hawaii.sys.i.kyoto-u.ac.jp/~junich-h/

Abstract. Several researchers have recently proposed alternative estimation methods of Boltzmann machines (BMs) beyond the standard maximum likelihood framework. Examples are the contrastive divergence or the ratio matching, and also a rather classic pseudolikelihood method. With a loss of statistical efficiency, alternative methods can often speed-up the computation and/or simplify the implementation. In this article, as an extreme of this direction, we show the parameter estimation of BMs can be done even with a closed-form estimator, by recasting the problem into linear regression. We confirm our estimator can actually approach the true parameter as the sample size increases, while the convergence can be slow, by a simple simulation experiment.

1 Introduction

Boltzmann machine [1] (BM) is a classic model of neural computation to learn multivariate probability distributions from data. It is closely related to the Ising model in statistical physics, or to the Markov random field (MRF) often used in image analysis. Although being classical, BMs and its related models have still been an active research target (e.g., [5]), yielding many important applications.

Several authors have recently proposed new estimation methods of BMs beyond the standard maximum likelihood (ML) framework [4,7,8,12]. As is well known, the major difficulty of ML in BMs is its high computational burden caused by intractable calculation of the normalization constant. An another difficulty is that, while not often considered, the implementation can become quite involved due to the need for "inference" about each variable even in the "fully-visible" cases (i.e, no hidden variables). Recent advanced techniques of probabilistic inference is not so easy for non-experts to effectively implement. Alternative methods are useful in that they can often speed-up the computation as well as simplify the implementation, by admitting a loss of statistical efficiency as ML performs.

In this article, as an extreme of such a direction to seek light and convenient estimation schemes, we present a *closed-form* estimator of BMs, while all of the existing approaches normally do not have such explicit forms. Although being closed-form do not necessarily mean its computational efficiency, it will still

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 951–959, 2009.

significantly simplifies the implementation. We present the basic idea and show empirical results, focusing on fully-visible BMs.

2 Brief Review of Boltzmann Machines

Let $\mathbf{x} = (x_1, x_2, \ldots, x_m)^\top \in \{-1, 1\}^m$ be an m-dimensional binary random vector. BM defines a probability distribution of $\mathbf{x}$ typically in the form

$$p(\mathbf{x} \mid \mathbf{W}, \mathbf{b}) = \frac{1}{Z(\mathbf{W}, \mathbf{b})} \exp\left(\frac{1}{2}\mathbf{x}^\top \mathbf{W}\mathbf{x} + \mathbf{b}^\top \mathbf{x}\right), \tag{1}$$

where

$$Z(\mathbf{W}, \mathbf{b}) = \sum_{\mathbf{x} \in \{-1,1\}^m} \exp\left(\frac{1}{2}\mathbf{x}^\top \mathbf{W}\mathbf{x} + \mathbf{b}^\top \mathbf{x}\right) \tag{2}$$

is called a partition function. The weight matrix $\mathbf{W}$ is usually assumed as symmetric ($w_{ij} = w_{ji}$) and having zero diagonal elements ($w_{ii} = 0$). Then, Eq. (1) can be written as

$$p(\mathbf{x} \mid \mathbf{W}, \mathbf{b}) = \frac{1}{Z(\mathbf{W}, \mathbf{b})} \exp\left(\sum_{i>j} w_{ij} x_i x_j + \sum_i b_i x_i\right). \tag{3}$$

2.1 Maximum Likelihood

Given a dataset $\mathcal{D} = \{\mathbf{x}^n\}_{n=1}^N$, where n is the sample index, the log-likelihood function of parameters $\mathbf{W}$ and $\mathbf{b}$ is given by

$$L(\mathbf{W}, \mathbf{b}) = N\left(\sum_{i>j} w_{ij} \langle x_i x_j \rangle_{\rho(\mathbf{x})} + \sum_i b_i \langle x_i \rangle_{\rho(\mathbf{x})} - \log Z(\mathbf{W}, \mathbf{b})\right) \tag{4a}$$

where $\langle \cdot \rangle_{p(\mathbf{x})}$ denote the expectation with respect to $p(\mathbf{x})$ and $\rho(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^N \mathbb{I}(\mathbf{x} = \mathbf{x}^n)$ is an empirical distribution of $\mathbf{x}$[1]. The gradient of the log-likelihood is also given by

$$\frac{\partial}{\partial w_{ij}} L(\mathbf{W}, \mathbf{b}) = N\left(\langle x_i x_j \rangle_{\rho(\mathbf{x})} - \langle x_i x_j \rangle_{p(\mathbf{x}|\mathbf{W},\mathbf{b})}\right), \tag{5a}$$

$$\frac{\partial}{\partial b_i} L(\mathbf{W}, \mathbf{b}) = N\left(\langle x_i \rangle_{\rho(\mathbf{x})} - \langle x_i \rangle_{p(\mathbf{x}|\mathbf{W},\mathbf{b})}\right). \tag{5b}$$

This involves intractable summations over 2^m states of $\mathbf{x}$ and thus some approximation is needed.

[1] $\mathbb{I}(\cdot)$ is the indicator function taking 1 (0) if the argument is true (false).

A standard approach to this approximation is to use a Markov chain Monte Carlo method, typically the Gibbs sampling, which iteratively generates a sample from the conditional distribution:

$$p(x_i \mid \mathbf{x}_{\setminus i}, \mathbf{W}, \mathbf{b}) = \frac{\exp(x_i \xi_i)}{\sum_{z_i \in \{-1,1\}} \exp(z_i \xi_i)}, \quad \xi_i = \sum_{j \neq i} w_{ij} x_j + b_i, \tag{6}$$

where $\mathbf{x}_{\setminus i}$ is all the variables except for x_i, and the summation $\sum_{j \neq i}$ is taken over $j \in \{1, 2, \ldots, i-1, i+1, \ldots, m\}$. A "naive" mean-field approximation [13] of the gradient is also popular in which the single Gibbs sampling step is replaced by taking the conditional expectation

$$\bar{x}_i = E\left[x_i \mid \mathbf{x}_{\setminus i}\right] = \frac{\exp(\xi_i) - \exp(-\xi_i)}{\exp(\xi_i) + \exp(-\xi_i)} = \tanh(\xi_i). \tag{7}$$

After converged to equilibria $\bar{x}_i^*$'s, the expectations $\langle x_i \rangle_{p(\mathbf{x}|\mathbf{W},\mathbf{b})}$ and $\langle x_i x_j \rangle_{p(\mathbf{x}|\mathbf{W},\mathbf{b})}$ are approximated respectively by $\bar{x}_i^*$ and $\bar{x}_i^* \bar{x}_j^*$. More advanced mean-field techniques can also be used [9]. However, they require the MCMC or mean-field iterations to be converged in each single step of gradient evaluation[2]; approximate ML estimations are thus essentially heavy in computation.

2.2 Maximum Pseudolikelihood and Related Methods

Recently, several alternatives to the ML have been proposed. Such methods include the contrastive divergence (CD) [4], the ratio matching [8] (a discrete analog of the score matching [6] of the same author), multi-conditional learning [12], etc. They are appealing both in their computational efficiency and also in the relative ease of implementation.

Although their forms are different from each other, many existing approaches in this direction can be regarded as variants of, or at least being closely related to, the rather old principle of maximum pseudolikelihood (MPL) [2], or more generally the composite likelihood [11,15,14], as recently suggested by several authors [7,10]. Thus, we here only describe MPL.

The estimator by MPL is obtained by maximizing the log-pseudolikelihood. In the case of BMs, this is given by

$$L_p(\mathbf{W}, \mathbf{b}) = N \sum_{i=1}^{m} \left\langle \log p(x_i \mid \mathbf{x}_{\setminus i}, \mathbf{W}, \mathbf{b}) \right\rangle_{\rho(\mathbf{x})} \tag{8a}$$

$$= N \sum_{i=1}^{m} \Big(\langle x_i \xi_i \rangle_\rho - \langle \log \cosh(\xi_i) \rangle_\rho + \log 2 \Big), \tag{8b}$$

where $\langle x_i \xi_i \rangle_\rho = \sum_{j \in \mathcal{V}_{\setminus i}} w_{ij} \langle x_i x_j \rangle_\rho + b_i \langle x_i \rangle_\rho$. The gradient of the log-pseudolikelihood is then given by

[2] But see [9] for efficient special cases.

$$\frac{\partial}{\partial w_{ij}} L_P(\mathbf{W}, \mathbf{b}) = N\left(\langle x_i x_j \rangle_{\rho(\mathbf{x})} - \left(\langle \bar{x}_i x_j \rangle_\rho + \langle x_i \bar{x}_j \rangle_\rho\right)/2\right), \tag{9a}$$

$$\frac{\partial}{\partial b_i} L_P(\mathbf{W}, \mathbf{b}) = N\left(\langle x_i \rangle_\rho - \langle \bar{x}_i \rangle_\rho\right). \tag{9b}$$

This can be easily evaluated without any iterative computations. Hyvärinen [7] has showed the consistency of the MPL estimator in fully-visible BMs.

3 Proposed Method

Let $p(\mathbf{x})$ be the target (true) distribution of discrete random vector $\mathbf{x}$. We denote a model as $q_{\boldsymbol{\theta}}(\mathbf{x})$ which has the form

$$q_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{\widetilde{q}_{\boldsymbol{\theta}}(\mathbf{x})}{Z(\boldsymbol{\theta})}, \tag{10}$$

with a partition function $Z(\boldsymbol{\theta}) = \sum_{\mathbf{x}} \widetilde{q}_{\boldsymbol{\theta}}(\mathbf{x})$. Our objective is to estimate such a $\boldsymbol{\theta}$ that the model $q_{\boldsymbol{\theta}}(\mathbf{x})$ becomes a good approximation to the target $p(\mathbf{x})$.

3.1 Basic Idea

Our starting point is a simple observation about the sufficient condition to attain $q_{\boldsymbol{\theta}}(\mathbf{x}) = p(\mathbf{x})$: This only requires $\widetilde{q}_{\boldsymbol{\theta}}(\mathbf{x})$ to be proportional to $p(\mathbf{x})$, since the normalization condition $\sum_{\mathbf{x}} p(\mathbf{x}) = \sum_{\mathbf{x}} q_{\boldsymbol{\theta}}(\mathbf{x}) = 1$ then automatically makes $q_{\boldsymbol{\theta}} = p$. Equivalently, this means that any function $r(\mathbf{x})$ that depends only on the ratio $\widetilde{q}_{\boldsymbol{\theta}}(\mathbf{x})/p(\mathbf{x})$ should be constant for any $\mathbf{x}$.

We now assume $r(\mathbf{x}) = \log(\widetilde{q}_{\boldsymbol{\theta}}(\mathbf{x})/p(\mathbf{x}))$. Note that $r(\mathbf{x})$ is a function of $\mathbf{x}$ and thus is a random variable under $\mathbf{x} \sim p(\mathbf{x})$. Then, $r(\mathbf{x})$ becomes constant over the support of $p(\mathbf{x})$ (i.e., region of non-zero probabilities) if and only if its variance is zero. If the support is sufficiently large within the overall domain of $\mathbf{x}$, so as for the zero-variance condition to offer enough constraints on $r(\mathbf{x})$, then $r(\mathbf{x})$ becomes constant for overall domain of $\mathbf{x}$. A lower variance also implies the function is nearly constant in a similar way. Thus, one can possibly measure the discrepancy between the two distributions by

$$\mathrm{D}[p(\mathbf{x}) \| q_{\boldsymbol{\theta}}(\mathbf{x})] \equiv \left\langle \left(\log \frac{p(\mathbf{x})}{\widetilde{q}_{\boldsymbol{\theta}}(\mathbf{x})} - \left\langle \log \frac{p(\mathbf{x})}{\widetilde{q}_{\boldsymbol{\theta}}(\mathbf{x})} \right\rangle_p \right)^2 \right\rangle_p = \mathbb{V}_p\left[\log \frac{p(\mathbf{x})}{\widetilde{q}_{\boldsymbol{\theta}}(\mathbf{x})} \right], \tag{11}$$

where $\mathbb{V}_p[\cdot]$ is the variance under the distribution p. This measure is completely independent of $Z(\boldsymbol{\theta})$. We also note the above quantity is also equal to $\mathbb{V}_p[\log(p(\mathbf{x})/q_{\boldsymbol{\theta}}(\mathbf{x}))]$, since the variance is unchanged by adding a constant $\log Z(\boldsymbol{\theta})$.

To estimate $\boldsymbol{\theta}$ from data based on this measure, we replace $p(\mathbf{x})$ by the empirical $\rho(\mathbf{x})$ as usual, and then minimize it with respect to $\boldsymbol{\theta}$. One may concern about the existence of $\log \rho(\mathbf{x})$ in the above measure, which may cause troublesome log-of-zeros. However, this is not true because $\rho(\mathbf{x})$, i.e., the normalized counts, is evaluated only at sampled points (where ρ is always positive)[3].

Although $\rho(\mathbf{x})$ with small sample sizes usually contains zero-probabilities and thus a zero variance does not necessarily assure $q_{\boldsymbol{\theta}}(\mathbf{x}) = \rho(\mathbf{x})$. However, in the asymptotic limit $(N \to \infty)$, $\rho(\mathbf{x})$ has a sufficiently large support if the target $p(\mathbf{x})$ is actually so; the zero variance then implies $q_{\boldsymbol{\theta}}(\boldsymbol{x}) = \rho(\mathbf{x}) = p(\mathbf{x})$. Thus, we can expect this scheme will produce a consistent estimator of the BM parameter, when the target is actually given in the BM family of finite parameter values (where $p(\mathbf{x}) > 0$ always holds).

3.2 Closed-Form Estimator for BMs

Now consider that the model $q_{\boldsymbol{\theta}}(\mathbf{x})$ is a BM. For convenience, we use the exponential family formulation of BM:

$$\widetilde{q}_{\boldsymbol{\theta}}(\mathbf{x}) = \exp\left(\boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x})\right), \tag{12}$$

where

$$\boldsymbol{\theta} = \begin{pmatrix} \mathbf{b} \\ \operatorname{vech}(\mathbf{W}) \end{pmatrix} \quad \text{and} \quad \boldsymbol{\phi}(\mathbf{x}) = \begin{pmatrix} \mathbf{x} \\ \operatorname{vech}\left(\mathbf{x}\mathbf{x}^\top\right) \end{pmatrix} \tag{13}$$

are the canonical parameter and sufficient statistics, respectively. Here we denoted $\operatorname{vech}(\mathbf{A})$ as a column vector that stacks the lower-triangle off-diagonal elements of matrix $\mathbf{A}$.

Let $h(\mathbf{x}) = \log \rho(\mathbf{x})$, then the empirical version of Eq. (11) is given by

$$\begin{aligned} \mathrm{D}[\rho(\mathbf{x}) \| q_{\boldsymbol{\theta}}(\mathbf{x})] &= \mathbb{V}_\rho\left[h(\mathbf{x}) - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x})\right] & \text{(14a)} \\ &= \mathbb{V}_\rho\left[\boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x})\right] - 2\mathbb{C}_\rho\left[h(\mathbf{x}), \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x})\right] + \text{const.} & \text{(14b)} \\ &= \boldsymbol{\theta}^T \mathbf{H} \boldsymbol{\theta} - 2\boldsymbol{\theta}^\top \mathbf{u} + \text{const.}, & \text{(14c)} \end{aligned}$$

where $\mathbb{C}_p[\cdot,\cdot]$ denote the covariance under p and we defined

$$\begin{aligned} \mathbf{H} &= \left\langle \boldsymbol{\phi}(\mathbf{x})\boldsymbol{\phi}(\mathbf{x})^\top \right\rangle_\rho - \langle \boldsymbol{\phi}(\mathbf{x}) \rangle_\rho \langle \boldsymbol{\phi}(\mathbf{x}) \rangle_\rho^\top, & \text{(15a)} \\ \mathbf{u} &= \langle \boldsymbol{\phi}(\mathbf{x}) h(\mathbf{x}) \rangle_\rho - \langle \boldsymbol{\phi}(\mathbf{x}) \rangle_\rho \langle h(\mathbf{x}) \rangle_\rho. & \text{(15b)} \end{aligned}$$

Note that $\mathbf{H}$ is the covariance matrix of $\boldsymbol{\phi}(\mathbf{x})$ and $\mathbf{u}$ a vector of cross-covariances between $\boldsymbol{\phi}(\mathbf{x})$ and $h(\mathbf{x})$. The estimator of $\boldsymbol{\theta}$, i.e., the minimum of Eq. (14c), is thus given in a closed-form:

$$\hat{\boldsymbol{\theta}} = \mathbf{H}^{-1}\mathbf{u}, \tag{16}$$

by solving the stationary condition $\nabla_{\boldsymbol{\theta}} \mathrm{D} = 2(\mathbf{H}\boldsymbol{\theta} - \mathbf{u}) = 0$.

[3] However, if $\mathbf{x}$ is continuous (with the summation replaced by integration), the variance diverges since $\rho(\mathbf{x}^n)$ usually have infinite point density.

3.3 A View from the Least Square Regression

The derivation above has a simple interpretation as a least square problem. Let a scalar β be an auxiliary parameter to form a linear model:

$$y^n = g_{\boldsymbol{\theta}}(\mathbf{x}^n) + \beta + \epsilon^n, \tag{17}$$

where we denoted $y^n = h(\mathbf{x}^n) = \log \rho(\mathbf{x}^n)$ and $g_{\boldsymbol{\theta}}(\mathbf{x}) = \log \widetilde{q}_{\boldsymbol{\theta}}(\mathbf{x})$; ϵ^n is the residual error. With a straightforward calculation, we can see the mean squared error:

$$\left\langle (h(\mathbf{x}) - g_{\boldsymbol{\theta}}(\mathbf{x}) - \beta)^2 \right\rangle_{\rho} = \frac{1}{N} \sum_{n=1}^{N} (y^n - g_{\boldsymbol{\theta}}(\mathbf{x}^n) - \beta)^2 , \tag{18}$$

is equal to Eq. (11) by plugging in the least square estimator of β, that is, $\hat{\beta} = \frac{1}{N} \sum_{n=1}^{N} (y^n - g_{\boldsymbol{\theta}}(\mathbf{x}^n))$.

Thus, our estimation scheme is basically solving a least square problem in which a target y^n is artificially created from an input $\mathbf{x}^n$ by using $h(\mathbf{x}) = \log \rho(\mathbf{x})$ as a "true" target function; $g_{\boldsymbol{\theta}}(\mathbf{x})$ is the regression function with a bias parameter β. In the exponential family case, since $g_{\boldsymbol{\theta}}(\mathbf{x}) = \log \widetilde{q}_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{x})$, the problem is thus simplified into a linear least square for which a closed-form estimator exists.

3.4 Implementation Issue

We conveniently implement our estimator by utilizing the above interpretation as linear regression, instead of directly calculating Eq. (16). The linear model in the vector notation is

$$\mathbf{y} = \mathbf{X}\boldsymbol{\omega} + \boldsymbol{\epsilon}, \tag{19}$$

where $\mathbf{y} = (y^1, y^2, \ldots, y^N)^\top$ and $\boldsymbol{\epsilon} = (\epsilon^1, \epsilon^2, \ldots, \epsilon^N)^\top$ are the target and residual vectors. The design matrix $\mathbf{X}$ and coefficient vector $\boldsymbol{\omega}$ are defined respectively as

$$\mathbf{X} = \begin{pmatrix} 1 & \boldsymbol{\phi}(\mathbf{x}^1)^\top \\ 1 & \boldsymbol{\phi}(\mathbf{x}^2)^\top \\ \vdots & \vdots \\ 1 & \boldsymbol{\phi}(\mathbf{x}^N)^\top \end{pmatrix}, \quad \boldsymbol{\omega} = \begin{pmatrix} \beta \\ \boldsymbol{\theta} \end{pmatrix}. \tag{20}$$

By solving the least square problem $\hat{\boldsymbol{\omega}} = \operatorname{argmin}_{\boldsymbol{\omega}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\omega}\|^2$ with any standard techniques, we can obtain $\hat{\boldsymbol{\theta}}$ by simply discarding the first element of $\hat{\boldsymbol{\omega}}$ ($= \hat{\beta}$, while this may be useful as an approximate value of $-\log Z(\hat{\boldsymbol{\theta}})$)[4].

[4] The explicit form of $\hat{\omega}$ is $\hat{\boldsymbol{\omega}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y}$; one can easily confirm the result is equivalent to Eq. (16), by utilizing a formula of block matrix inversion [3].

In practice, however, the design matrix $\mathbf{X}$ often has a degenerate column rank so that a unique solution $\hat{\boldsymbol{\omega}}$ does not exist. As in the standard recipe of linear regression, we typically employ the pseudo-inverse $\mathbf{X}^{\dagger}$ to obtain the solution of minimum Euclid norm by $\hat{\boldsymbol{\omega}} = \mathbf{X}^{\dagger}\mathbf{y}$, or explicitly add a quadratic regularizer to the cost function to have a "ridge" estimator $\hat{\boldsymbol{\omega}} = \left(\mathbf{X}^{\top}\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^{\top}\mathbf{y}$, with an appropriate setting of $\lambda > 0$. In the experiment of Sec. 4, we used the former scheme.

4 Simulation Results

We conducted an simulation experiment to empirically examine the basic property of our method in a similar setting to that used in [7]. Our method was compared to both ML and MPL, as well as a standard MAP (*maximum-a-posteriori*) estimation, i.e., a penalized ML with a quadratic regularizer[5]. The data were artificially generated from BMs with $m = 5$ and 8 which were small enough to allow exact sampling of data as well as to compute ML (and MAP) estimates exactly. The sample size N was varied as 10, 50, 100, 500, 1000, 2000, 4000, 8000, 16000, 32000 and 64000, in each of which ten different datasets were created by randomly setting the true parameter $\{w_{ij}\}$ and $\{b_i\}$ each generated from a Gaussian $\mathrm{N}(0, 0.5^2)$.

Figure 1 shows the mean squared errors (MSEs) between the estimated and true parameters, where each line shows the median of the ten runs with each errorbar indicating the upper and lower quartiles. In large sample sizes (say, ≥ 100 for $m = 5$ or ≥ 500 for $m = 8$), we can confirm the MSE of our estimator monotonically decreases as the sample size increases. The error was comparable to the others when $m = 5$, while was larger when $m = 8$. In contrast, in smaller sample sizes, our estimator initially exhibited lower errors than both ML and MPL, but the error did not decrease until enough amount of samples were collected. Similar improvements in small sample sizes were found in the MAP results.

5 Discussion

In this article, we have presented a new method to estimate the parameter of BMs, based on an idea to minimize the variance of $\log(\rho(\mathbf{x})/\widetilde{q}_{\boldsymbol{\theta}}(\mathbf{x}))$. This can also be interpreted as a least square problem. This interpretation offered a convenient implementation of our method. However, if we start from this regression view, the squared error may not be the best to explain the difference between the empirical and true log-probability functions. Extension to other error functions will be an interesting direction of future study, while the solution may have no closed-form instead.

Although our estimation scheme is probably new, its practical availability would be currently questionable. As demonstrated, the speed of convergence to (possibly) the true parameter can be quite slow even in comparison to MPL.

[5] This maximize $L(\boldsymbol{\theta}) - \eta\|\boldsymbol{\theta}\|^2$ where η is set here at 0.01.

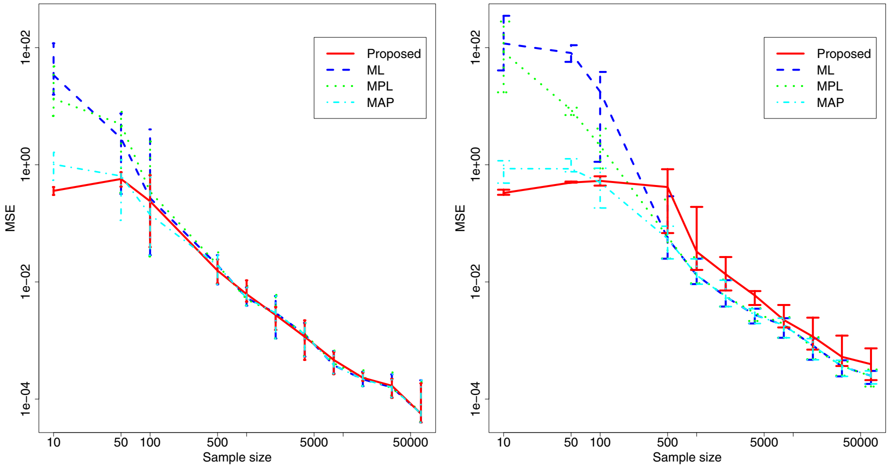


Fig. 1. MSE for $m = 5$ (left) and $m = 8$ (right). Both axes are in logarithmic scales.

From the regression view, we conjecture that a sparse “input” distribution $\rho(\mathbf{x})$ could cause this slowness. Then, the success of our method will be limited to such cases that there are sufficiently large data points as well as the target distribution is not too sparse, while the complementary cases are of major concern in practice. Relaxing this limitation by, for example, introducing a suitable regularization to our method is remained for future study.

References

1. Ackley, D., Hinton, G., Sejnowski, T.: A learning algorithm for Boltzmann machines. Cognitive Science 9(1), 147–169 (1985)
2. Besag, J.: Statistical analysis of non-lattice data. Statistician 24, 179–195 (1975)
3. Harville, D.A.: Matrix Algebra From A Statistician’s Perspective. Springer, Heidelberg (1997)
4. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural Computation 14(8), 1771–1800 (2002)
5. Hinton, G.E., Salakhutdinov, P.R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (2006)
6. Hyvärinen, A.: Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research 6, 695–709 (2005)
7. Hyvärinen, A.: Consistency of pseudolikelihood estimation of fully visible Boltzmann machines. Neural Computation 18(10), 2283–2292 (2006)
8. Hyvärinen, A.: Some extensions of score matching. Computational Statistics & Data Analysis 51, 2499–2521 (2007)
9. Kappen, H., Rodriguez, F.: Effcient learning in Boltzmann machines using linear response theory. Neural Computation 10(5), 1137–1156 (1998)
10. Liang, P., Jordan, M.I.: An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In: Proceedings of the 25th international conference on Machine learning (ICML 2008), pp. 584–591 (2008)

11. Lindsay, B.G.: Composite likelihood methods. Contemporary Mathematics 80, 221–239 (1988)
12. McCallum, A., Pal, C., Druck, G., Wang, X.: Multi-conditional learning: Generative/discriminative training for clustering and classification. In: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2006), pp. 433–439 (2006)
13. Peterson, C., Anderson, J.: A mean field theory learning algorithm for neural networks. Complex Systems 58, 1486–1493 (1987)
14. Varin, C.: On composite marginal likelihoods. Advances in Statistical Analysis 92(1), 1–28 (2008)
15. Varin, C., Vidoni, P.: A note on composite likelihood inference and model selection. Biometrika 92(3), 519–528 (2005)

Incremental Learning in the Non-negative Matrix Factorization

Sven Rebhan[1], Waqas Sharif[2], and Julian Eggert[1]

[1] Honda Research Institute Europe GmbH
Carl-Legien-Strasse 30, 63073 Offenbach am Main, Germany
[2] Darmstadt University of Technology (Control Theory and Robotics Lab)
Landgraf-Georg-Strasse 4, 64283 Darmstadt, Germany

Abstract. The non-negative matrix factorization (NMF) is capable of factorizing strictly positive data into strictly positive activations and base vectors. In its standard form, the input data must be presented as a batch of data. This means the NMF is only able to represent the input space contained in this batch of data whereas it is not able to adapt to changes afterwards. In this paper we propose a method to overcome this limitation and to enable the NMF to incrementally and continously adapt to new data. The proposed algorithm is able to cover the (possibly growing) input space *without* putting further constraints on the algorithm. We show that using our method the NMF is able to approximate the dimensionality of a dataset and therefore is capable to determine the required number of base vectors automatically.

1 Introduction

The NMF has been introduced by Lee and Seung [1,2] as an unsupervised factorization method for decomposing multi-variant data under the constraint of non-negativity. By only allowing the additive combination of components, the method generated a parts-based representation. In its standard form, the NMF works as a batch algorithm, i.e. the whole dataset is presented at once and has to cover the desired input space entirely. Changes in that space can only be incorporated by restarting the learning process from scratch, using the new and the old data samples together to represent the new input space. This requires an enormous amount of memory and computational effort, because all data samples seen so far have to be stored, and the base vectors have to be recomputed once a new sample is presented. An additional drawback of the NMF and batch algorithms in general is that we have to specify the number of base vectors a priori. This poses a hard problem for many real-world datasets, because the intrinsic dimensionality of those datasets is not known.

Cao et al. developed an online variant of the NMF [3] trying to overcome the mentioned limitations. In their approach it is possible to add new data samples to the representation later, making it possible to adapt to temporally changing data. Nevertheless, it is necessary to put an orthogonality constraint on the learning process in order to obtain a proper representation of the input space.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 960–969, 2009.

Furthermore, both the initial number of vectors and the number of additionally available vectors for new learning cycles still have to be specified beforehand.

We propose a method to learn the available data sequentially and hereby to incrementally adapt the base vectors to represent the input space. By doing so, the algorithm is able to autonomously approximate the intrinsic dimensionality of the input data, rendering it unnecessary to specify the number of base vectors in advance. By assuming to only see one new data sample at a time we, contrary to [3], do not need to put further constraints on the learning process.

In Sect. 2 we briefly review the standard NMF algorithm and its update procedure, before we describe the idea of the incremental NMF (iNMF) in Sect. 3. Then we present our results (Sect. 4) using the bar dataset and the Essex face94 dataset [4]. Finally, we discuss the results and give an outlook on future work.

2 Standard NMF Review

The NMF algorithm as described by Lee and Seung [1,2] is based on a distance measure between the input dataset $\mathbf{V}$ and the reconstruction $\mathbf{R}$. We here focus on the Euclidian distance measure

$$F(\mathbf{W},\mathbf{H}) = ||\mathbf{V} - \mathbf{R}||^2 \ , \tag{1}$$

where the reconstruction is calculated by the linear superposition of the base vectors $\mathbf{W}$ weighted with their corresponding activation $\mathbf{H}$

$$\mathbf{V} \approx \mathbf{R} = \mathbf{W}\mathbf{H} \ . \tag{2}$$

Lee and Seung have shown that one can derive the following multiplicative update rules for $\mathbf{W}$ and $\mathbf{H}$ to minimize the error function of Eq. 1[1]

$$\mathbf{H} \leftarrow \mathbf{H} \odot \frac{\mathbf{W}^T\mathbf{V}}{\mathbf{W}^T\mathbf{R}} \tag{3}$$

$$\mathbf{W} \leftarrow \mathbf{W} \odot \frac{\mathbf{V}\mathbf{H}^T}{\mathbf{R}\mathbf{H}^T} \tag{4}$$

By alternately performing the update steps for $\mathbf{H}$ and $\mathbf{W}$, the algorithm is able to find at least a locally optimal solution for Eq. 1 (see [2]). The final update schema for the NMF reads as follows:

1. Calculate the reconstruction according to Eq. 2.
2. Update the activations using Eq. 3.
3. Calculate the reconstruction according to Eq. 2.
4. Update the base vectors using Eq. 4.
5. Repeat step 1 to 4 until convergence.

[1] $\odot$ and $\div$ denote componentwise operations: $\mathbf{a} \odot \mathbf{b} := A_i \cdot B_i, \ \forall i$.

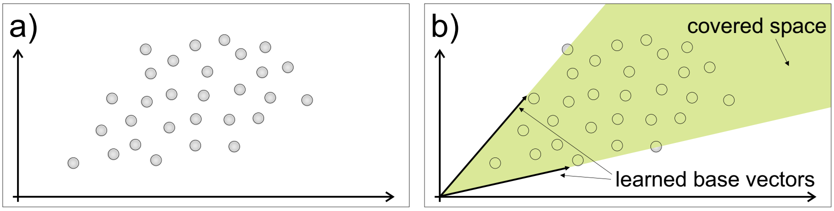


Fig. 1. While learning a dataset a), the base vectors are adapted to cover at least the space spanned by the data samples b)

The matrix $\mathbf{V}$ contains a set of data samples $\mathbf{v}_i$ as column vectors, representing the input space that should be covered by the base vectors (see Fig. 1a for an example). The available number of base vectors is mainly responsible for the learning result and must be defined a priori. If the number of the base vectors is greater or equal to the intrinsic dimensionality of the data, the NMF learns base vectors that cover at least the space spanned by the data samples (see Fig. 1b). If their number is too low, the NMF will not be able to derive sensible base vectors and will end up with a high reconstruction error for all input data. Another problem related to the NMF and batch algorithms in general is that it is impossible to adjust the representation of the input space to subsequent and new information, after the learning process took place.

3 Incremental NMF Algorithm (iNMF)

To overcome the mentioned limitations, we propose to incrementally extend the representation of the input space and enable the algorithm to add further base vectors if necessary. The learning process, as schematically described in Fig. 2, starts with only one base vector. In each learning cycle the incremental NMF(iNMF) only sees one data sample at a time. For the first data sample we perform a NMF learning (see section 2) with the single base vector. The resulting base vector, as shown in Fig. 2b, represents all points along its extension, lying on a straight line and hence covering a 1D-space. To start the next learning cycle using a new data sample $\mathbf{d}(t)$, we have to make sure that the already acquired information is preserved. Here we exploit the fact that the learned base vector represents all information about the previously seen data samples (see [5] and Eq. 2), which is the basic idea of our method. So all we need to do is to append the new data sample $\mathbf{d}(t)$ to the previously learned base vectors $\mathbf{W}(t-1)$ in order to obtain the new NMF input vector $\mathbf{V}(t)$

$$\mathbf{V}(t) = \{\mathbf{d}(t), \mathbf{W}(t-1)\} \ . \tag{5}$$

With the new $\mathbf{V}(t)$ we now perform a new learning cycle equivalent to the NMF update schema (see section 2). For the example shown in Fig. 2c, both the new sample and the previous base vector are already covered by the set of base vectors $\mathbf{W}(t-1)$, so there is no need for adaptation.

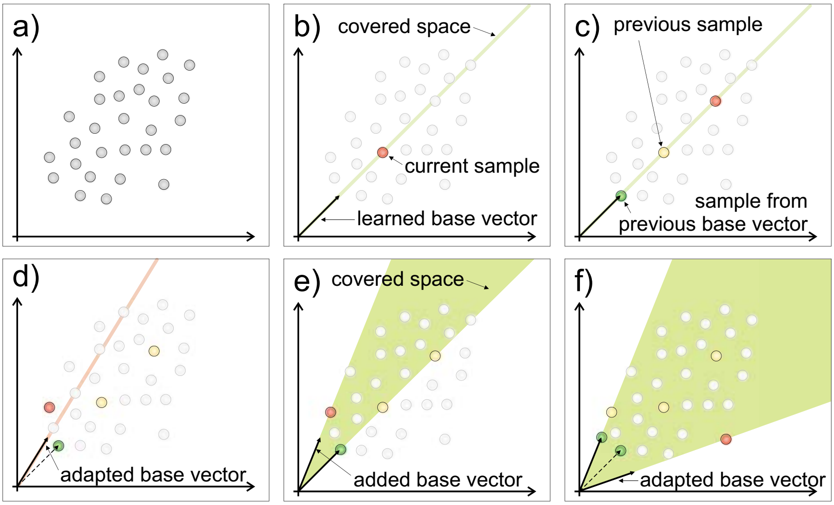


Fig. 2. The learning and adding of base vectors in the iNMF shown as: a) the available data samples. b) the learned base vector from the first data sample covering a 1D-space. c) a new data sample and the learned base vector as a virtual sample (containing the first sample shown). Both data points are inside the covered 1D-space, allowing for the reconstruction using a single base vector. d) the third data sample being outside the covered 1D-space. Even adapting the available base vector leads to a large reconstruction error. e) by adding and adapting a base vector both the virtual sample and the new data sample can be reconstructed. As a result a 2D-space is covered. f) by adapting the two base vectors a larger region of the 2D-space can be covered. Here the intrinsic number of dimensions (two) is found by the iNMF.

If we provide a data sample beyond the covered space, a reconstruction of both the new data sample and the previous base vector is not longer possible. In Fig. 2d this is due to the fact that the data samples span a 2D-space, but only a 1D-space can be covered with a single vector. Although the NMF tries to adapt the existing vector to minimize the reconstruction error $F(\mathbf{W}, \mathbf{H})$ (see Eq. 1), a high error will remain. We know that the NMF is able to reconstruct all data samples in an n-dimensional space if the number of base vectors is at least equal to the dimensionality of that space (see [5] and Fig. 2 for illustration). So a high reconstruction error indicates that the dimensionality of the space spanned by the data samples must be higher than the number of currently available base vectors. Because we provide the iNMF only with one new data sample at a time, we know that the dimensionality also must have increased by one and hence requires one additional base vector. By providing this vector it is possible to span a 2D-space and to reach a good reconstruction quality by repeating the NMF learning cycle (see Fig. 2e). The dataset used for our illustration (see Fig. 2a) is 2D, it will be possible to reconstruct all further data samples provided to the iNMF by adapting the two available base vectors as shown in Fig. 2f. By successively applying our method, the iNMF algorithm is able to approximate the dimensionality of the input data by itself. We can formulate the update schema for the iNMF in the following pseudo code:

```
1. Randomly initialize a single base vector $\mathbf{w}_0$ and set $\mathbf{W}(t-1) = \{\}$.
2. Take the next data sample $\mathbf{d}(t)$ and construct the input vector
   $\mathbf{V}(t) = \{\mathbf{d}(t), \mathbf{W}(t-1)\}$ as described in Eq. 5.
3. Set $\mathbf{W}(t) = \mathbf{W}(t-1)$ to initialize the base vectors.
4. Initialize the activation matrix $\mathbf{H}$ randomly.
5. Perform a NMF learning cycle:
   (a) Calculate the reconstruction according to Eq. 2.
   (b) Update the activations using Eq. 3.
   (c) Calculate the reconstruction according to Eq. 2.
   (d) Update the base vectors using Eq. 4.
   (e) Repeat step 5a to 5d until convergence.
6. If the reconstruction error $F(\mathbf{W}, \mathbf{H})$ (Eq. 1) is low, continue with step 2.
   Otherwise provide an additional base vector $\mathbf{W}(t) = \{\mathbf{W}(t-1), \mathbf{W}_{new}\}$ and
   a corresponding activation. Continue with step 4.
```

4 Results

4.1 Bar Dataset

The bar dataset we have used consists of 162 images with a size of 32x32 pixel. Each of the images is a superposition of up to four horizontal and vertical bars. A horizontal bar can be applied to four different horizontal positions; the vertical bar to four different vertical positions. Complete overlapping of two bars is not allowed. We permute the original dataset (see e.g. Fig 3) in order to destroy a potential order (from single bars to many bars) in the dataset and remove all images containing only a single bar. This prevents the iNMF from focusing on the

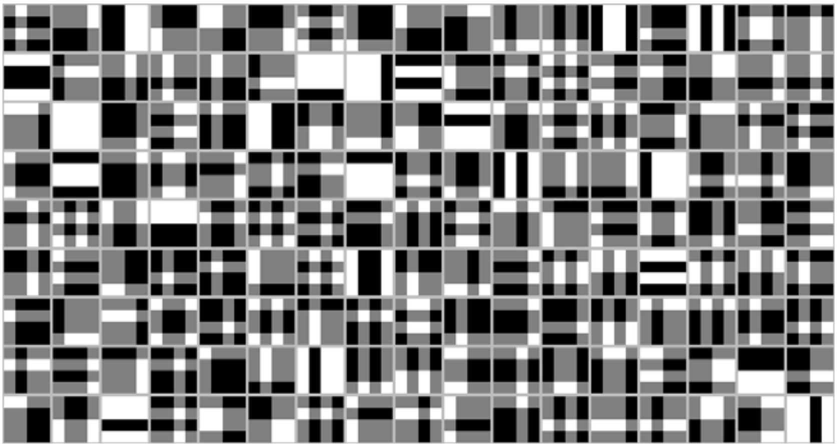


Fig. 3. Here you can see the 153 images of the permuted bar dataset

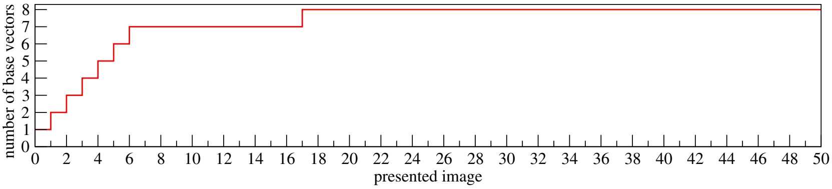


Fig. 4. After about 20 images, the iNMF reaches the final number of base vectors, which are adapted afterwards. Due to visibility reasons, we only show the first 50 images

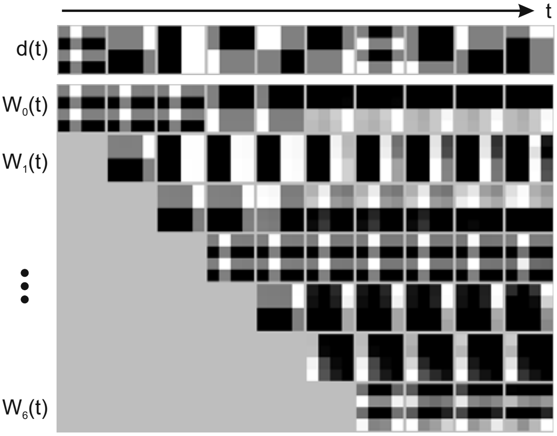


Fig. 5. While provided with the first ten images (top), the iNMF algorithm generates seven base vectors (bottom). The first seven input images are simply stored one-to-one in the base vectors. After this phase, the base vectors are adapted and become less complex and sparser.

trivial cases where solely the initial eight vectors are sufficient to learn the input space. Figure 4 shows the massive necessity to add base vectors within the first 7 images. During this phase the algorithm copies the data samples one-to-one into base vectors as shown in Fig. 5. Afterwards, the number of vectors stays constant for about 10 input images. Here, the reconstruction of the input data is achieved by adapting the already existing vectors and thus removing redundancy in the set of base vectors. This can be interpreted as a specialization of the vectors, resulting in a lower complexity, lower redundancy and higher sparsity.

If we compare the base vectors after presenting 20 images with the final base vectors (see Fig. 6), we see that the final shape of the base vectors has already emerged. Merely the amplitude of some pixels is different. So already at this early point the input space is covered to a very large degree. We have tested ten different permutations of the bar dataset and found that the final number of vectors is reached between the 9th and 76th image. This also implies that the dimensionality of the input dataset was correctly estimated. In all cases the iNMF algorithm is able to find the correct base vectors out of the randomly

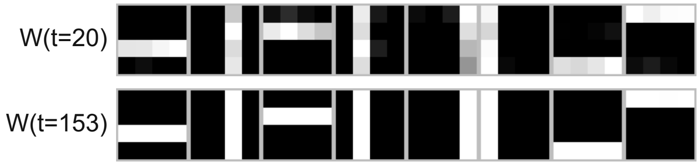


Fig. 6. A comparison between the vectors after 20 images (top) and the final vectors after 153 images (bottom) shows that the final shape of the base vectors has already emerged after 20 images

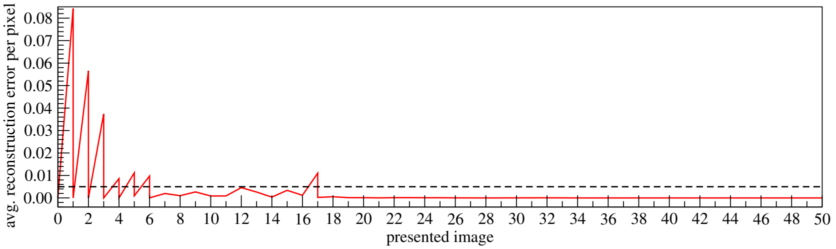


Fig. 7. The error function over the presented images shows characteristic peaks if the number of base vectors is insufficient. A vector is added if the error exceeds the threshold = 0.005 (dashed), increasing the reconstruction quality drastically. We only show the first 50 images here.

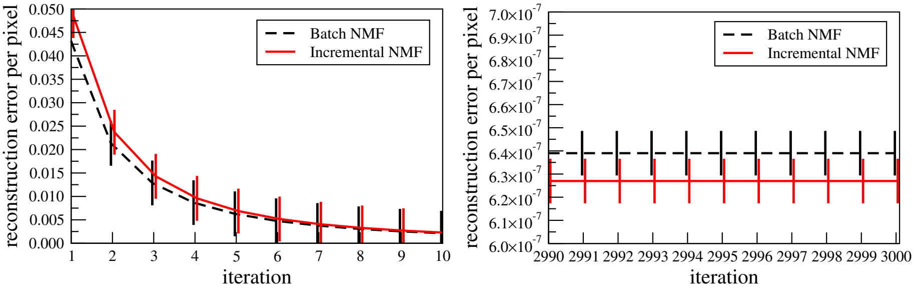


Fig. 8. Mean and standard deviation (scaled by 3 for visualization) in the first and last 10 steps of the bar dataset reconstruction. The iNMF and the NMF perform comparably good.

ordered samples that dot not contain the initial vectors. A typical development of the error function over the presented images is shown in Fig. 7. In the diagram one can see sharp rises of the reconstruction error, which are characteristic for an insufficient number of base vectors. If such a peak is detected, a new base vector will be added and we can observe a drastic drop in the reconstruction error. This results from the fact that the iNMF algorithm is now able to place the new base vector pointing into the direction of the current sample and thus leading to a nearly perfect reconstruction. In our case we have chosen a fixed threshold to detect the critical reconstruction error. The threshold itself can be varied in a large range (from 0.0001 to 0.01), assuring an easy handling.

4.2 Comparison to Batch NMF

To compare the iNMF to the NMF we apply both algorithms to the bar dataset (see Fig. 3). Afterwards we use the final base vectors of both algorithms to reconstruct the dataset to evaluate how much information is stored in the base vectors. During this phase we do not adapt the base vectors, but keep them fixed. Each algorithm is started with 10 randomly chosen initial activations. In Fig. 8 you can see the resulting mean and standard deviation comparison of the reconstruction quality between the NMF and the iNMF. As you can see, the incrementally learned base vectors of the iNMF exhibit a comparable reconstruction quality on the dataset with respect to the base vectors learned by the batch NMF algorithm.

4.3 Essex Face94 Dataset

The Essex face dataset [4] contains images of 152 individuals at 20 different postures with a size of 180 by 200 pixels. For our experiments we have only used a part of the dataset comprising 780 face images including 19 female and 20 male individuals at 20 different postures (see Fig. 9). For the test on the face dataset the same error threshold was chosen as for the bar data.

The effect of adding a base vector at the 260th image is shown as an example in Fig. 10. Here you can see a drastic improvement in the reconstruction quality after adding a new base vector. The older base vectors W_0 to W_3 are already very sparse and specialized for certain face regions. The new base vector first points directly in the direction of the current image. After a few iterations, the vector also gets less complex and sparser.

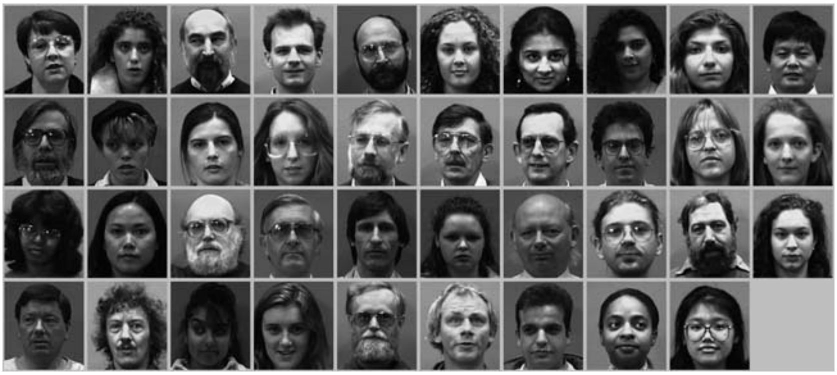


Fig. 9. The part of the Essex face94 dataset we have used includes the 39 individuals depicted here. For each individual 20 different postures are available.

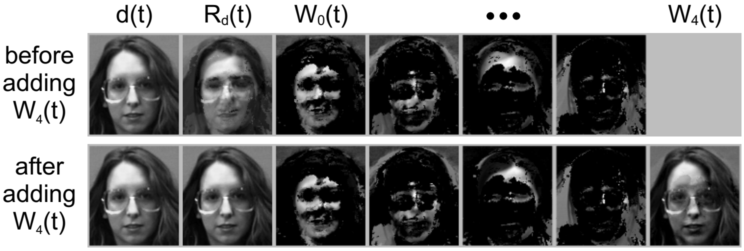


Fig. 10. The reconstruction error before adding a base vector (top) is very high. After adding the base vector $\mathbf{w}_4(t)$ (bottom), the reconstruction quality increases drastically.

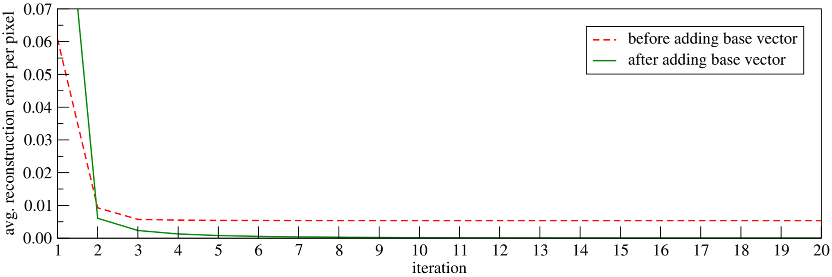


Fig. 11. The error for the insufficient number of base vectors stays high (dashed), whereas after adding a vector (solid) it drops to nearly zero

In Fig. 11 we show the error function before and after adding the base vector. Both curves show a similar development, but only with the additional vector a sufficiently good reconstruction quality can be reached, whereas before, the error could not fall below a certain value.

Figure 12 depicts the final six base vectors and six randomly selected data samples $\mathbf{d}$ with their corresponding reconstruction $\mathbf{R}$. It strikes that the vec-

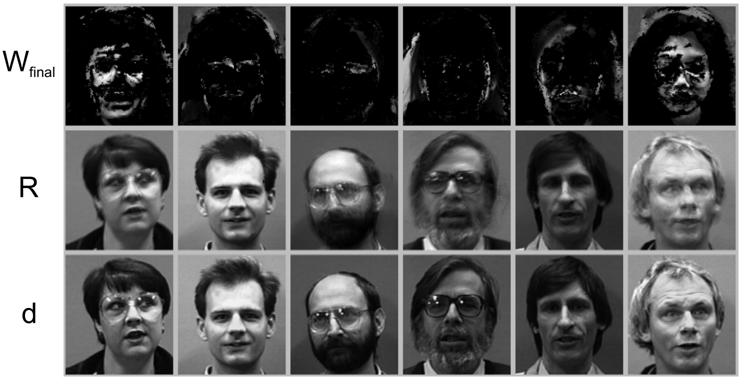


Fig. 12. Here you can see the final base vectors generated by the iNMF (top) and the reconstruction (middle) of selected faces (bottom)

tors represent ”parts” of a face. Furthermore, one can see that the information about the input space is preserved to a large degree in the base vectors, as the reconstructions show. In this case, the iNMF also finds the number of base vectors by itself, starting with only one vector.

5 Discussion

In this paper we have shown that the incremental NMF is able to handle both artificial and real-world data in a sequential manner. We exploit the fact that the base vectors themselves already represent all previously shown data samples. Starting from only one single base vector the iNMF decides autonomously if and when an additional base vector is required to reconstruct the input space. The presented algorithm overcomes the need to choose the number of base vectors a priori. This eases the use of the iNMF, especially if the intrinsic dimensionality of the dataset is not known, which is often the case for real-world data. Nevertheless the reconstruction quality of the iNMF is comparable to the NMF (see Sect. 4.2). Furthermore it is possible to perform both online computation and long-term adaptation and at the same time reduce the memory and computational requirements. These reduced requirements make the processing of huge real-world datasets possible in the first place. Since the proposed principle is not tailored to the NMF only, we propose to investigate ways to transfer the presented incremental schema to other factorization methods.

References

1. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature 401, 788–791 (1999)
2. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: NIPS, pp. 556–562 (2000)
3. Cao, B., Shen, D., Sun, J.-T., Wang, X., Yang, Q., Chen, Z.: Detect and track latent factors with online nonnegative matrix factorization. In: IJCAI, pp. 2689–2694 (2007)
4. University of Essex: Essex face94 dataset, http://cswww.essex.ac.uk/mv/allfaces/faces94.html
5. Donoho, D., Stodden, V.: When does non-negative matrix factorization give a correct decomposition into parts? In: NIPS (2003)

Contextual Behaviors and Internal Representations Acquired by Reinforcement Learning with a Recurrent Neural Network in a Continuous State and Action Space Task

Hiroki Utsunomiya and Katsunari Shibata

Oita University 700 Dannoharu Oita Japan
shibata@cc.oita-u.ac.jp

Abstract. For the progress in developing human-like intelligence in robots, autonomous and purposive learning of adaptive memory function is significant. The combination of reinforcement learning (RL) and recurrent neural network (RNN) seems promising for it. However, it has not been applied to a continuous state-action space task, nor has its internal representations been analyzed in depth. In this paper, in a continuous state-action space task, it is shown that a robot learned to memorize necessary information and to behave appropriately according to it even though no special technique other than RL and RNN was utilized. Three types of hidden neurons that seemed to contribute to remembering the necessary information were observed. Furthermore, by manipulate them, the robot changed its behavior as if the memorized information was forgotten or swapped. That makes us feel a potential towards the emergence of higher functions in this very simple learning system.

1 Introduction

It goes without saying that it is important to memorize useful information from time series of sensor signals and utilizing it when required. If such functions could be learned autonomously, it would be a great progress in developing human-like intelligence in robot. In reinforcement learning (RL) research, such functions have been discussed as one of the ways to solve POMDP (Partially Observable Markov Decision Problem) or perceptual aliasing problems. Many techniques have been proposed to solve this problem by introducing some memory system, and among them, utilizing a recurrent neural network (RNN) seems the most promising way because of its autonomous ability to extract important information effectively and to utilize it. After the proposition of the "Recurrent-Q" technique[1], several works have followed[2][3]. Most of them deal with tasks with a discrete state space. Against such a trend, there are some works which tackled to the continuous state space problems[4][5][6]. Onat et al.[4] showed that a dynamical system could be controlled without perceiving velocity information. In the work of Bakker et al[5], a robot learned to memorize and utilize important binary information to get a reward in a T-maze problem even though it had to

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 970–978, 2009.

make another decision before it made the decision that required the memorized information. They also showed that their method worked on an experiment using a real robot[6]. However, the action space is still discrete and the memorized information was not investigated in much depth.

In this paper, we applied RL with a RNN to a robot navigation task in which a robot has to choose one of two goals according to the flag signals perceived on a switch. The situation is similar to that in [5], but the exploration field is a two-dimensional free space, and the robot, goals, and switch are located randomly in the field at every episode. The state and action spaces are both continuous. Another big difference is that the processing system is as simple as just a RNN trained by RL, while in [5], special technique named "ARAVQ" was used to extract events in episodes. A significant point in this paper is that without employing any special techniques, interesting contextual behaviors are acquired through learning. In order to extract important events from episodes, some relevant criterion to judge the importance is usually introduced. However, when considering that RL has its criterion for the system optimization to increase its reward and decrease its punishment as much as possible, the criterion should be consistent in one system. We think that we should leave as much of robot process as possible to the autonomous learning by the combination of RL and NN. Furthermore, the hidden representation acquired through learning is observed, and an examination of the change in contextual behavior through manipulations of the outputs of hidden neurons allows us to investigate the properties of the memorized information in depth.

2 Learning System

The learning system is very simple. From sensors to motors, there is only one popular Elman-type recurrent neural network (RNN)[7] whose hidden outputs are fed back to the input layer at the next time step. It is trained by popular reinforcement learning (RL) technique, that is Actor-Critic with TD-Learning[8]. The present observation vector $\boldsymbol{x}_t$ is the network input. The output neurons are divided into one critic (state value) output and the other actor (motor commands) outputs. The motor command vector $\tilde{\boldsymbol{A}}_t$ is derived by adding an uniform random number vector $\boldsymbol{rnd}_t$ to the actor output $\boldsymbol{A}(\boldsymbol{x}_t)$. After the robot has moved according to $\tilde{\boldsymbol{A}}_t$, a new observation vector is obtained. After forward computation of the RNN for the new input signals $\boldsymbol{x}_{t+1}$, the training signal $C_{d,t}$ for the critic and $\boldsymbol{A}_{d,t}$ for the actor output vector for the previous time step are generated autonomously based on Temporal Difference learning as

$$C_{d,t} = C(\boldsymbol{x}_t) + \hat{r}_t = r_{t+1} + \gamma C(\boldsymbol{x}_{t+1}) \tag{1}$$

$$\boldsymbol{A}_{d,t} = \boldsymbol{A}(\boldsymbol{x}_t) + (\tilde{\boldsymbol{A}}_t - \boldsymbol{A}(\boldsymbol{x}_t))\hat{r}_t = \boldsymbol{A}(\boldsymbol{x}_t) + \boldsymbol{rnd}_t \cdot \hat{r}_t \tag{2}$$

$$\hat{r}_t = r_{t+1} + \gamma C(\boldsymbol{x}_{t+1}) - C(\boldsymbol{x}_t) \tag{3}$$

where r_t is a given reward, γ is a discount factor, $\hat{r}_t$ is the TD error, and C is the critic output. After one more forward computation for the input signals at the

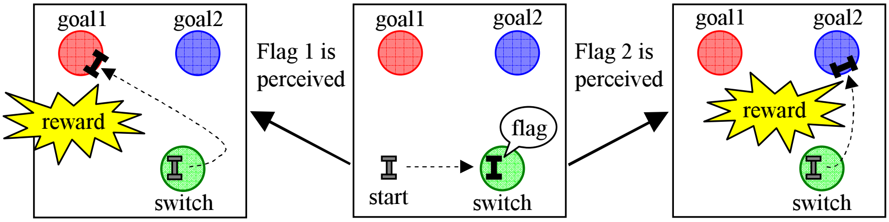


Fig. 1. The mission of the robot is to step on the switch at first and then to go to the correct goal according to the flag signals that can be perceived only on the switch

previous time step, the training signals are given to the corresponding output, and the network is trained by BPTT (Back Propagation Through Time)[9]. The output function used in the hidden and output neurons is the sigmoid function whose value ranges from -0.5 to 0.5, and the training signals are bounded from -0.4 to 0.4. To adjust the offset between actual critic and network output, they are shifted up or down by 0.5. The network output 0.0 corresponds to critic 0.5.

3 Experimental Results

In this experiment, a task in which a robot requires contextual behavior was employed. As shown in Fig.1, on a two dimensional 15 × 15 continuous, flat square space of arbitrary distance units, the robot, one switch and two goals are located randomly at the beginning of every episode. They do not overlap with each other. The goals and switch are circles of radius 0.5. At first, the robot has to step on a switch before it approaches to a goal. Only when the robot is on the switch, it can perceive the two flag signals, and one of them chosen randomly is 1 and the other is 0. When the robot is not on the switch, they are always 0. At each episode, the robot can know from the flag signals which goal it should go to.

As shown in Fig.2, the observation vector has 13 elements in total. 5 signals represent the distances to the objects and wall. 6 signals represent the angles to the objects. 2 signals represent the flag information. A transformation using the exponential function on the distance signals magnifies the range for small distances. The signals are input into the RNN. Before learning, the robot does not know the meaning of the switch, goals, or flag signals. The robot has two wheels. It can rotate its right and left wheels separately, and so the robot can move any directions except sideways. Each element of $\boldsymbol{A}_t$ multiplied by 1.25 indicates the distance that each wheel moves in one time step. The value is bounded from -0.5 to 0.5. The interval between two wheels is 0.32. The robot can move up to 0.5 forward or backward, and can rotate up to 180 degrees in one time step. Whether the robot steps on a goal or switch is judged actually by whether the center of the robot is on the area of goal or switch.

The task is episodic. Before each episode, all the hidden outputs are reset to 0.0, and the critic value after reaching a goal is 0.0. Only when the robot steps on

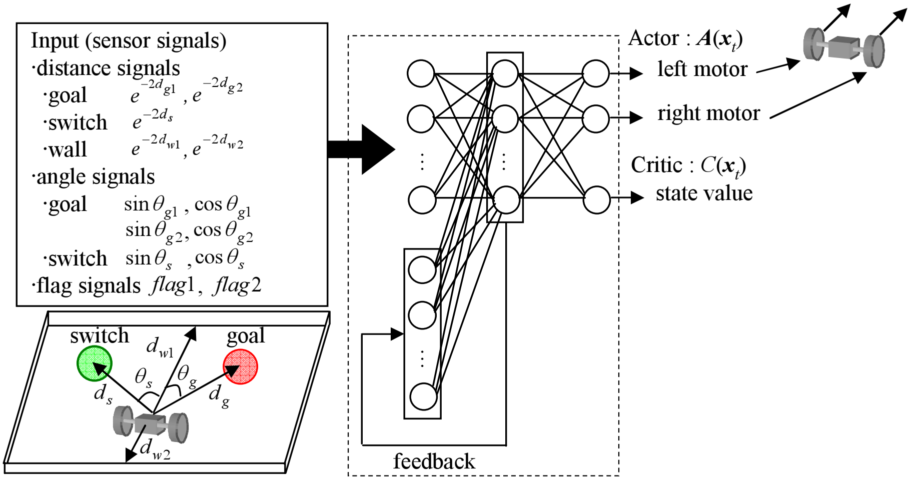

Input (sensor signals)
·distance signals
·goal $e^{-2d_{g1}}, e^{-2d_{g2}}$
·switch e^{-2d_s}
·wall $e^{-2d_{w1}}, e^{-2d_{w2}}$
·angle signals
·goal $\sin\theta_{g1}, \cos\theta_{g1}$
$\sin\theta_{g2}, \cos\theta_{g2}$
·switch $\sin\theta_s, \cos\theta_s$
·flag signals *flag*1, *flag*2
switch
goal
d_{w1}
θ_s
θ_g
d_s
d_g
d_{w2}
Actor : $\boldsymbol{A}(\boldsymbol{x}_t)$
left motor
right motor
Critic : $C(\boldsymbol{x}_t)$
state value
feedback

Fig. 2. Input and output signals of the recurrent neural network

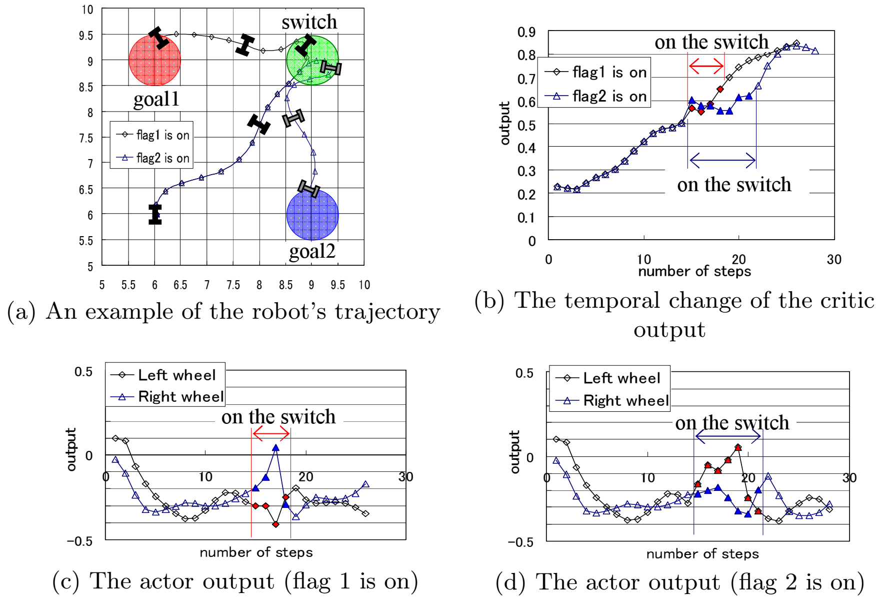
the correct goal after stepping on the switch it can get a reward $r = 0.9$, and the episode terminates. However, when the robot goes to the incorrect goal or goes to the correct goal before stepping on the switch, it does not get any reward, but a penalty of $r = -0.1$, and the episode does not terminate. Moreover, if the robot bumps a wall, it also gets a penalty of $r = -0.1$.

The task is divided into 3 levels, and the learning becomes increasingly difficult as shown in Table 1. Initial location range means all the objects are located in this area. Exploration range means the range of the random numbers in $\boldsymbol{rnd}$ that is added to the actor output as trial and error factors. If the condition that the robot reaches the correct goal within 50 time steps is satisfied for 100 successive episodes, the robot moves to the next level. The learning terminates when the robot is in the level 3 and the condition is satisfied for 100 episodes.

The RNN has three layers and 50 hidden neurons. The maximum time steps traced back through time for BPTT is 20. The discount factor γ is 0.96. The initial weight for each hidden-output connection is 0.0 and that for each non-feedback input-hidden connection is chosen randomly from -1.0 to 1.0. For the self-feedback connections, the initial weight is 4.0, while that for the other

Table 1. Incremental learning difficulties. x indicates the number of episodes in the level.

	Level1	**Level2**	**Level3**
Initial location range	2×2	3×3	4×4
Exploration range	$\pm\frac{1}{2}e^{-5\times10^{-5}x}$	$\pm\frac{2}{5}e^{-5\times10^{-5}x}$	$\pm\frac{3}{10}e^{-5\times10^{-5}x}$
Upper bound of episode length	10000	5000	3000



(a) An example of the robot's trajectory

(b) The temporal change of the critic output

(c) The actor output (flag 1 is on)

(d) The actor output (flag 2 is on)

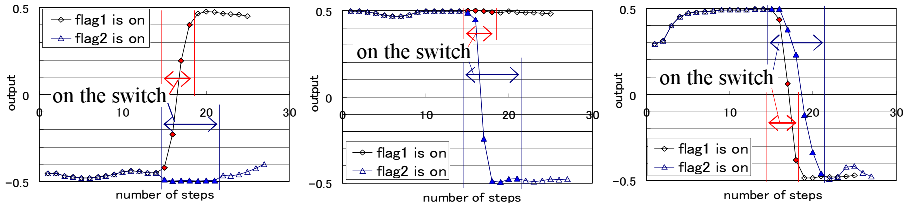
Fig. 3. A sample of the learning result for both flag states after learning

feedback connection is 0.0, which makes the learning of memory function easy. The learning constant is 0.05 for the feedback connections, and 0.2 for the others. More than 10 runs with different random number sequences were performed, but the learning results were similar to each other. One of them is introduced in the following.

Learning terminates after 31131 episodes. The robot can go to the switch at first, and then it can approach the correct goal corresponding to the flag signals perceived only at the switch for any initial locations such as Fig.3 (a). In Fig.3 (b), the critic output increases in both cases, but in Fig.3 (c) and (d), we can see big differences in the actor outputs especially on the switch that make the difference in the direction of the robot's approach after stepping on it.

4 Contextual Behaviors and Hidden States: Tests and Observations

How the robot remembers the information of flag signals in the RNN after learning is analyzed here. First, we observed the temporal change of each hidden neuron during one episode for several switch and goal configurations. There are three types of hidden neurons that are considered to contribute to remembering the flag signals after leaving the switch. Fig.4 shows the response of a hidden neuron for each type to each flag state in the example of Fig.3. The type 1 neurons changed its output only when the flag 1 was on, and the type 2 neurons changed its output only when the flag 2 was on. On the other hand, the type 3



(a) Type 1
(responds only to flag 1)

(b) Type 2
(responds only to flag 2)

(c) Type 3
(responds to both flags)

Fig. 4. Three types of hidden neurons that seem to contribute to keeping the flag state

neurons changed its output at the switch for both cases. There are 7 type 1, 4 type 2, and 4 type 3 neurons out of the total of 50 hidden neurons.

4.1 Manipulating of Hidden State during an Episode

In order to make sure that these neurons remember the flag state, the hidden neuron outputs were manipulated. Before the robot reached the switch, the outputs of type 1 hidden neurons that respond only to the flag 1 were stored, and after leaving the switch, the outputs of type 1 hidden neurons were reset to those stored before the switch.

When the flag 1 was on, the robot returned to and stepped on the switch again and then approached the goal 1 as shown in Fig.5 (a). However, when the flag 2 was on, the value of the hidden neuron did not change noticeably after the resetting. The robot's trajectory did not seem to change, and it went to the goal 2 without returning to the switch. When the outputs of type 2 hidden neurons that respond only to the flag 2 were stored and reset, the robot returned to the switch when the flag 2 was on, but when the flag 1 was on, it approached to the goal 1 without returning to the switch. It is thought that the division of roles in memory among hidden neurons was acquired through RL.

Next, in only one of the type 1 hidden neurons, the value was stored before stepping on the switch and was reset to the stored one afterwards. Then, soon after the reset, the hidden neuron retrieved the value just before the reset, and the robot's trajectory does not change noticeably. It is thought that the robot memorized the information about the flag with a distributed representation among some neurons, and the function of associative memory using the recurrent structure emerged through RL.

In the third test, the flag 1 was on at first. After the robot left the switch, the outputs of all the three types of hidden neurons were stored. Then the robot was taken back to the initial location with all hidden neurons being reset to 0.0 and began to move again. However, this time, the flag 2 was on and the flag 1 was off on the switch. As shown in Fig.6(a), after stepping on the switch, it began to approach the goal 2 because the robot perceived the flag 2 on the switch, but after that, the outputs of all the three types of hidden neurons were reset to the stored ones when the flag 1 had been on. After that, the robot changed its

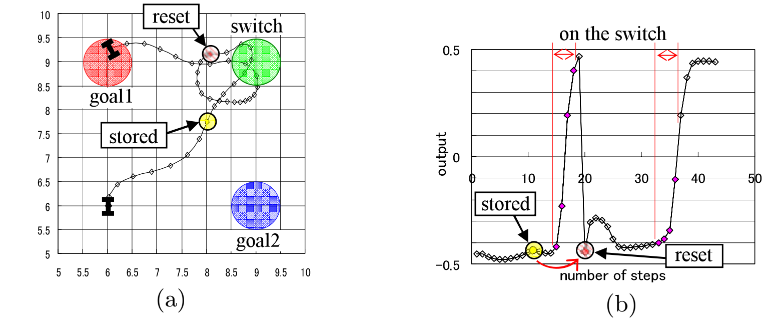


Fig. 5. (a) Robot's trajectory when the output of type 1 hidden neurons were reset to those stored before stepping on the switch and (b) output change of one of the type 1 hidden neurons for the case of (a)

direction to the goal 1. However, since that was not the correct goal, no reward was given and the episode did not terminate there. Surprisingly, as shown in Fig. 6(b), the robot then returned to the switch again and approached the goal 2 that is the correct one.

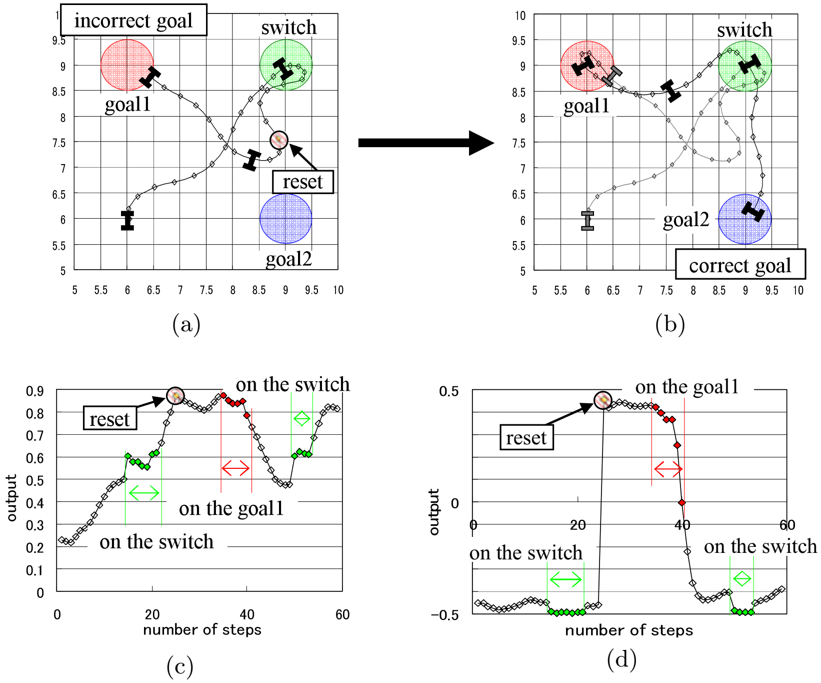


Fig. 6. Robot's trajectory (a) until it arrived at the incorrect goal and (b) after the arrival, when the flag 2 was on and the hidden neuron outputs were reset to those stored when the flag 1 had been on. Output change of (c) critic and (d) a type 1 hidden neuron for the case of both (a) and (b).

As shown in Fig.6(c) and (d), it is interesting that when the episode did not terminate on the goal 1, the outputs of the critic and type 1 hidden neurons that respond only to the flag 1 decreased suddenly. In the robot's experiences, when it stepped on the switch beforehand and went to the correct goal, the episode always terminated with a reward. When the episode did not terminate, it had missed to step on the switch in advance or had arrived to the incorrect goal. It is suggested that due to the generalization from such experiences, the robot misunderstood that it had forgotten to step on the switch even though it actually had. Such behaviors seem very human-like. The intelligent behaviors seem to be controlled depending on the abstract state that is constracted by memorizing the flag signals in the RNN based on understanding of their importance. We suppose it shows the potential toward the emergence of higher functions that the proposed very simple learning system has.

5 Conclusion

A robot with a very simple learning system consisting of a recurrent neural network trained using reinforcement learning without any other special techniques learned a continuous state-action space task that requires a memory function. After learning, it was observed that replacement of the hidden states in the middle of one episode lead to sudden change of the contextual behavior. It was also observed that no termination of one episode caused a misunderstanding in the robot, which behaved as if it had forgotten to take the necessary action, and returned to perform the action again. We think that the learning system is so simple and general that it can be applied widely to many tasks. The observed interesting behaviors suggest that the learning system is very simple but promising towards the emergence of higher functions.

Acknowledgments. This research was supported by JSPS Grant in-Aid for Scientific Research ♯15300064 and ♯19300070.

References

1. Lin, L., Mitchell, T.M.: Memory Approaches to Reinforcement Learning In Non-Markovian Domains. Technical Report CMU-CS-TR-92-138, CMU, Computer Science (1992)
2. Onat, A., Kita, H., Nishikawa, Y.: Recurrent Neural Networks for Reinforcement Learning: Architecture, Learning Algorithms and Internal Representation. In: Proc. of IJCNN 1998, pp. 2010–2015 (1998)
3. Mizutani, E., Dreyfus, S.E.: Totally Model-Free Reinforcement Learning by Actor-Critic Elman Network in Non-Markovian Domains. In: Proc. of IJCNN 1998, pp. 2016–2021 (1998)
4. Onat, A., Kita, H., Nishikawa, Y.: Q-Learning with Recurrent Neural Networks as a Controller for the Inverted Pendulum Problem. In: Proc. of ICONIP 1998, pp. 837–840 (1998)

5. Bakker, B., Linaker, F., Schmidhuber, J.: Reinforcement Learning in Partially Observable Mobile Robot Domains Using Unsupervised Event Extraction. In: Proc. of IROS 2002, vol. 1, pp. 938–943 (2002)
6. Bakker, B., Zhumatiy, V., Gruener, G., Schmidhuber, J.: A Robot that Reinforcement-Learns to Identify and Memorize Important Previous Observations. In: Proc. of IROS 2003, pp. 430–435 (2003)
7. Elman, J.L.: Finding structure in time. Cognitive Science 14, 179–211 (1990)
8. Barto, A.G., Sutton, R.S., Anderson, W.: Neuronlike adaptive elements can solve difficult learning control problems. IEEE Trans. on Systems, Man, and Cybernetics 13(5), 834–846 (1983)
9. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In: Rumelhart, D.E., McClelland, J.L., Group, P.R. (eds.) Parallel distributed processing. MIT Press, Cambridge (1986)

Improving the Quality of EEG Data in Patients with Alzheimers Disease Using ICA

François-Benoit Vialatte[1], Jordi Solé-Casals[2], Monique Maurice[1], Charles Latchoumane[3], Nigel Hudson[4], Sunil Wimalaratna[5], Jaeseung Jeong[3], and Andrzej Cichocki[1]

[1] Riken BSI, Lab. ABSP, Wako-Shi, Japan
fvialatte@brain.riken.jp
[2] University of Vic, Signal Processing Group, Vic, Spain
[3] KAIST, Dept of Bio and Brain Engineering, Daejeon, South Korea
[4] Derriford Hospital, Dept of Neurophysiology, Plymouth, UK
[5] Radcliffe Infirmary, Dept of Neurology, Oxford, UK

Abstract. Does Independent Component Analysis (ICA) denature EEG signals? We applied ICA to two groups of subjects (mild Alzheimer patients and control subjects). The aim of this study was to examine whether or not the ICA method can reduce both group differences and within-subject variability. We found that ICA diminished Leave-One-Out root mean square error (RMSE) of validation (from 0.32 to 0.28), indicative of the reduction of group difference. More interestingly, ICA reduced the inter-subject variability within each group ($\sigma = 2.54$ in the δ range before ICA, $\sigma = 1.56$ after, Bartlett p = 0.046 after Bonferroni correction). Additionally, we present a method to limit the impact of human error ($\simeq$ 13.8%, with 75.6% inter-cleaner agreement) during ICA cleaning, and reduce human bias. These findings suggests the novel usefulness of ICA in clinical EEG in Alzheimer's disease for reduction of subject variability.

1 Introduction

Independent component analysis (ICA) is a method for recovering underlying signals from linear mixtures of those signals. Independent component analysis (ICA) is especially useful to reject EEG artifacts [1], exploiting statistical independent criteria to separate EEG sources, which allows to remove artifacts and clean EEG [2,3,4]. Therefore, it has been used for analysis of various physiological time series including the EEG. However, whether the ICA alters the EEG distributions or not is unclear. In addition to instrumental noise and environmental noise, movement and other physiological noise (ocular, electromyographic, electrodermal, electrovascular, and respiratory signals) may interfere with the EEG in the form of artifacts. Artifacts in the EEG can be defined as any potential difference due to an extra-cerebral source [5]. Particularly, muscle artifacts are especially problematic, because they can appear in EEG patterns which are very hard to differentiate from the EEG signals: the frequency range of muscle artifacts and the EEG overlap to a high degree [6].

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 979–986, 2009.

The aim of this study was to address the significant question: is it a valid approach to improve EEG signals using Independent component analysis (ICA)? We investigated the effects of ICA cleaning on a database recorded from patients with Alzheimer's disease (mild AD, early stage) and healthy control subjects. It is of importance to investigate the possible application of ICA to AD, because the early diagnosis of AD using EEGs is very critical [7]. Previous studies used ICA to enhance differences between control subjects and Alzheimer patients, but the effect on EEG quality was only assessed by estimating the group separation (see *e.g.* [3,8,9] - which is not sufficient to prove that ICA preserved EEG data. In this study, we used the ICA to show the following results:

1. Human performance displayed a high variability. We used a simple method to combine signals cleaned without concertizing by three scientist, which provided us with an optimally cleaned database (in respect to an Euclidean distance).
2. With proper precautions, ICA cleaning did not denature EEG signals.

Using the simple cleaning rules introduced in this study, it shall be possible to design a semi-automatic method to improve EEG quality.

2 Methods

Computations were done with Matalb (The MathWorks, Inc.), ICA cleaning was performed using ICALAB ver. 3 with automatic sorting of independent components [10].

2.1 EEG Data - Patients with MildAD

These data were obtained using a strict protocol from Derriford Hospital, Plymouth, U.K. and had been collected using normal hospital practices [11]. EEGs were recorded during a resting period with various states: awake, drowsy, alert and resting states with eyes closed and open. All recording sessions and experiments proceeded after obtaining the informed consent of the subjects or the caregivers and were approved by local institutional ethics committees. EEG dataset is composed of 24 healthy control subjects (age: 69.4+11.5 years old; 10 males) and 17 patients with mild AD (age: 77.6+10.0 years old; 9 males). The patient group underwent full battery of cognitive tests (Mini Mental State Examination, Rey Auditory Verbal Learning Test, Benton Visual Retention Test, and memory recall tests). The two groups are not perfectly age-matched, which might pose bias later on, but it was shown that no major effect was found due to this disparity [11]. The EEG time series were recorded using 19 electrodes disposed according to Maudsley system, similar to the 10-20 international system, at a sampling frequency of 128 Hz. EEGs were band-pass filtered with digital 2nd order Butterworth filter (forward and reverse filtering) between 0.5 and 30 Hz (a sampling rate of 128 Hz means that frequencies above 25 Hz cannot be reliably studied [12]).

2.2 Independent Component Analysis

Blind Source Separation (BSS) consists in recovering a set of unknown sources from their observed mixture $\mathbf{x}$. The linear and instantaneous models of BSS can be formulated as:

$$\mathbf{x} = \mathbf{A}\mathbf{s}, \tag{1}$$

where $\mathbf{s}$ represents a data matrix having as rows the observed signals, and $\mathbf{A}$ is the mixing matrix. According to the currently prevailing view of EEG signal processing, a signal can be modeled as a linear mixture of a finite number of brain sources, with additive noise(see *e.g.* [2,3,4]). Therefore, blind source separation techniques can be used advantageously for decomposing raw EEG data to brain signal subspace and noise subspace. If sources are supposed to be independent, then BSS can be called ICA.

The Second-Order Blind Identification (SOBI) algorithm is a well-known blind source separation (BSS) method for source signals with temporal structures and distinct spectra (AR processes). It already proved to be useful in many biomedical applications. A weight adjusted version of SOBI was suggested in [14]. SOBI jointly (approximately) diagonalizes time-delayed covariance matrices for many time delays. However, SOBI algorithm does not specify how many and which time delays to choose. An efficient weight adjusted variant of SOBI called IWA-SOBI [13,15] was recently developped to solve this problem. The original weight adjusted SOBI used a standard AJD (Approximative Joint Diagonalization) algorithm.IWASOBI uses instead an AJD based on family of WEDGE[1] algorithms [13]. For IWASOBI the number of jointly diagonalized covariance matrices can be relatively low in comparison to the standard SOBI while performance can be considerably higher. This algorithm allows reliable separation of 100+ sources with temporal structure (autoregressive sources) in order of seconds. In our experiments we used the IWASOBI algorithm implemented in ICALAB ver.3 [10].

2.3 Cleaning Rules

Three EEG researchers visually inspected EEGs, and chose the least corrupted (artifact-clean) continuous 20 sec interval of each recording for the analysis. Each trial was then decomposed using ICA. Sources were ordered using a kurtosis measure, and the researchers cleared up to seven sources per trial corresponding to artifacts (eye movements, EMG corruption, EKG, etc), using three criteria (see Fig.1):

1. Abnormal scalp distribution of the reconstructed channels (only a few electrodes contribute to the source, with an isolated topography)
2. Abnormal wave shape (drifts, eye blinks, sharp waves, etc.)
3. Source of abnormally high amplitude ($\geq 100\ \mu$V)

We have focused our attention mainly on the smallest and largest values of kurtosis (i.e. a measure of sparsity and distance to Gaussianity), which are more

[1] Weighted Exhaustive Diagonalization using Gauss itEration.

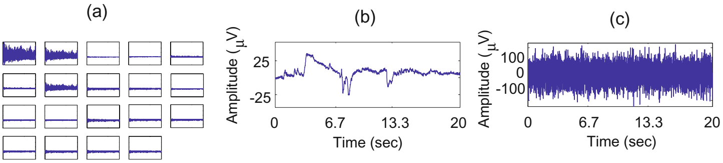


Fig. 1. Examples of artifacts. (a) abnormal scalp distribution of the reconstructed channels; (b) abnormal wave shape; (c) source of abnormally high amplitude.

likely to be representative of artifacts. After this step, the remaining sources were back-projected onto the scalp, yielding an artifact clean data.

2.4 Optimal Combination of Three Cleanings

Each of the three researchers cleaned, with the above rules, raw data from all subjects (EEG from mildAD patients or control subjects) s_n with $n \in [1-41]$. This produced three cleaned databases: $\mathbf{D_1}, \mathbf{D_2}$ and $\mathbf{D_3}$. Our objective was to combine efficiently these $\mathbf{D_i}$ databases into one optimal database $\mathbf{\Omega}$. Each database $\mathbf{D_i}$ includes cleaned EEG[2] $c_{i,n}$ with $n \in [1-41]$. We computed for all $c_{i,n}$ the averaged Fourier relative power $\mathbf{\Phi}(i,n,f)$ over all E electrodes:

$$\mathbf{\Phi}(i,n,f) = \frac{1}{E}\sum_{e=1}^{E} \mathbf{\Phi}_{i,n}(e,f). \tag{2}$$

Where $\mathbf{\Phi}_{i,n}(e,f)$ is the Fourier relative power of the electrode e with frequencies f in $\mathbf{F} = [1-25]$ Hz, for the cleaned EEG $c_{i,n}$. Fourier power was computed using the Welch method (1 sec. Hanning windows with 50% overlap). We then computed for each pair of cleaned EEG $(c_{i,n}, c_{j,n})$ the Euclidian distance $\mathbf{\Delta}$:

$$\mathbf{\Delta}(n,i,j) = \sqrt{\sum_{f in \mathbf{F}} [\mathbf{\Phi}(c_{i,n},f) - \mathbf{\Phi}(c_{j,n},f)]}. \tag{3}$$

This distance is symmetric, with values in $\mathbb{R}^+$, and $c_{i,n} \simeq c_{j,n} \Rightarrow \mathbf{\Delta}(n,i,j) \simeq 0$. $\mathbf{\Delta}$ was used to evaluate significant differences between cleanings. To this end we used a Monte-Carlo approach to estimate the distribution of randomly matched surrogate data. The surrogate database was constituted of $r = 10000$ draws of randomly matched data from the three cleaned databases. We computed the r distances $\mathbf{\Delta}(r,1,2)$ of each pair, which allowed us to estimate the distribution of non-matched data. We fixed the significance threshold $\tau = 1.1\%$, as the fifth percentile of this distribution (if $\mathbf{\Delta}(n,i,j) \leq \tau$, we reject the hypothesis of having a significant difference between cleaners i and j for subject s_n with a probability $p = 0.05$[3]).

[2] Corresponding to the raw data of s_n: for each subject s_n exist three cleaned EEG, $c_{1,n}, c_{2,n}$, and $c_{3,n}$.

[3] A probability $p = 0.01$ corresponds to a threshold $\tau = 0.89\%$.

For each of subject $s_n, n \in [1-41]$, we iteratively selected the best candidate $\omega_n \in \mathbf{\Omega}$ within the corresponding cleaned EEG triplet $[c_{1,n}, c_{2,n}, c_{3,n}]$ using the following decision rule:

1. Equivalent case: if $\forall(c_{i,n}, c_{j,n}), \mathbf{\Delta}(n,i,j) < \tau$, we select randomly the cleaned EEG ω_n.
2. Consensus case: if $\forall(c_{i,n}, c_{j,n})$, $\exists$ only $(c_{u,n}, c_{v,n}), \mathbf{\Delta}(n,u,v) \geq \tau$ (or conversely $\mathbf{\Delta}(n,v,u) \geq \tau$), we select the remaining cleaned EEG $\omega_n = c_{w,n}$ with $w \ni [u,v]$ ($c_{w,n}$ is a consensus of the two others cleaned EEG).
3. Error case: if $\exists u, \forall i, \mathbf{\Delta}(n,u,i) \geq \tau$, we select randomly $\omega_n = c_{v,n}$ with $v \neq u$ ($c_{u,n}$ is identified as an error).
4. Reject case: if $\forall(c_{i,n}, c_{j,n}), \mathbf{\Delta}(n,i,j) \geq \tau$, the subject s_n has to be rejected.

3 Results

3.1 Variability between Cleaners

The cleaned data were distributed in the four above categories as follows: 75.6% equivalent cases, 17.1% consensus cases, 7.3% error cases and 0% reject cases[4]. In other words, we could clean 100% of the database without rejection. From these numbers we can also obtain an approximate estimation of variability between human cleaners:

- Inter-cleaner agreement $\simeq$ 75.6%
- Human error rate for one isolated cleaner $\simeq$ 13.8%.
- Expected cleaning error % between two persons $\simeq$ 10.6%.

This indicates that EEG data containing a significant number of subjects, cleaned by only one or two persons, is not so reliable.

Table 1. Mann-Whitney Zscore before and after ICA, for each frequency range (comparing mildAD patients *vs.* control subjects). Higher absolute value of z-score means that p-value is lower (i.e. that data is better separated).

Data	δ (1-4 Hz)	θ (4-8 Hz)	α (8-12 Hz)	β (12-25 Hz)
Before ICA	3.68	2.83	-4.39	-3.78
After ICA	3.76	4.82	-4.58	-4.42

3.2 Group Differences

Mann-Whitney z-scores were estimated before and after ICA cleaning. We observed, for all frequency ranges, that the differences between mildAD patients

[4] This result is off course dependant on the threshold τ.

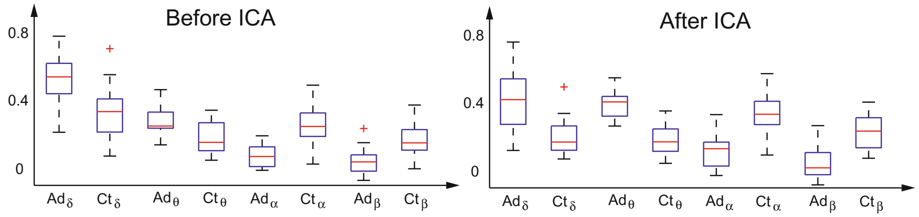


Fig. 2. Boxplot of Fourier relative power before and after ICA. Central line represents the median, dashed lines covers inter-quartile range, cross are outliers. ICA magnified the group differences.

and control subjects were magnified by ICA (see Table 1 and Fig.2). We aggregated the Fourier power into five regions (frontal, temporal left and right, central and occipital). Linear discriminant analysis was applied before and after ICA cleaning. We estimated Leave-One-Out root mean square error (RMSE) of validation: it dropped from 0.32 to 0.28 (training RMSE dropped from 0.30 to 0.26). This shows that the topography of EEG relative powers after ICA cleaning was more consistent than those before cleaning.

3.3 Inter-subject Variability

After ICA cleaning, we examine if EEG was not denatured. This can be observed through the distribution of Fourier power in the two groups: did the subjects of control and patient group resembled each-other more after ICA, or instead did they differ more (in this case, EEG is denatured).

We evaluated the inter-subject variability in both groups, by comparing their variance using a Bartlett test for homoscedasticity (before *vs.* after ICA). Variance was never shown to increase (even without Bonferroni correction), instead

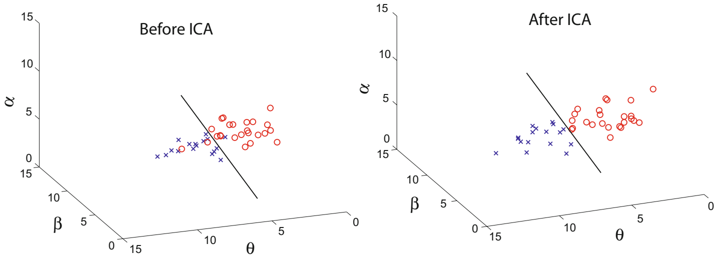


Fig. 3. Inter-subject variability of all subjects before and after ICA. Each control subject (circle) and mildAD patient (cross) are plotted a spatial representation of their Fourier power in the three lower frequency ranges - δ (1-4 Hz), θ (4-8 Hz) and α (8-12 Hz). Not only were subjects regrouped in their respective classes by ICA, but separation between the two groups was improved.

a significant decrease was found in the δ range for the control group (standard deviation $\sigma = 2.54$ before ICA, $\sigma = 1.56$ after, Bartlett p = 0.046 after Bonferroni correction). This decrease in inter-subject variability is also found generally: ICA subjects became closer to their group's mass centers (see Fig.3).

4 Discussion and Conclusion

We found that ICA cleaning improved the separation between mildAD patients and control subjects (confirming previous studies [4,8,9]). Moreover, we found a more consistent topographical distribution of EEG power after ICA. We could conjecture two possibilities concerning the effect of ICA on EEG quality:

1. Fourier power distribution changed after ICA, the variability in at least one group increased (*i.e.*, the variance of Fourier power increased in one group). In this case, ICA cleaning denatured EEG signals, as the data quality was lowered.
2. Fourier power distribution changed after ICA, the variability did not increase, and eventually decreased (i.e., the variance of Fourier power remained stable in both groups or decreased). In this case, ICA cleaning improved the quality of EEG signals.

We found the second situation, as a variance decrease in the control group was the only significant effect. We can therefore conclude that ICA cleaning does not denature EEG signals, and instead improves their quality. This results was however obtained with a combined cleaning done by three independant persons. Without such combination, human error might compensate with the benefits of ICA (error rate $\simeq 13.8\%$). Other ways to combine cleaning obtained from several persons could be imagined. We also attempted to average spectrograms of each database: *i.e.* averaging each subject's three occurrences instead of selecting one occurrence. This led to very similar results, EEG data was not denatured. However, we believe that averaging does not prevent the intrusion of seriously flawed data, whereas our selection approach is more preemptive.

These results suggests a new direction for EEG studies using ICA: developping organized methodologies of cleaning, *i.e.* coordinated semi-automatic methods of EEG cleaning. In other words, instead of only developing new algorithms, we should endeavor to find markers of artifacts and semi-automatic methods of cleaning. An ideal toolbox should provide synthetic information about each ICA sources (indicating for instance a pre-diagnostic of anomalies); human cleaners would then perform a cleaning based on this information (a semi-automatic cleaning).

Acknowledgments

J. Solé-Casals acknowledges support from the Ministerio de Educación y Ciencia of Spain (grant TEC2007-61535/TCM), and from the Universitat de Vic (grant R0912). Monique Maurice is supported by the JSPS fellowship P–08811.

References

1. Ille, N., Berg, P., Scherg, M.: Artifact correction of the ongoing eeg using spatial filters based on artifact and brain signal topographies. Journal of clinical neurophysiology 19(2), 113–124 (2002)
2. Jung, T.P., Makeig, S., Humphries, C., Lee, T.-W., McKeown, M.J., Iragui, V., Sejnowski, T.J.: Removing electroencephalographic artifacts by blind source separation. Psychophysiology 37, 163–178 (2000)
3. Cichocki, A.: Blind Signal Processing Methods for Analyzing Multichannel Brain Signals. International Journal of Bioelectromagnetism 6(1) (2004)
4. Cichocki, A., Shishkin, S.L., Musha, T., Leonowicz, Z., Asada, T., Kurachi, T.: EEG filtering based on blind source separation (BSS) for early detection of Alzheimer"s disease. Clinical Neurophysiology 116(3), 729–737 (2006)
5. Anderer, P., Roberts, S., Schlogl, A., Gruber, G., Klosch, G., Herrmann, W., Rappelsberger, P., Filz, O., Barbanoj, M.J., Dorffner, G., Saletu, B.: Artifact processing in computerized analysis of sleep eeg - a review. Neuropsychobiology 40(3), 150–157 (1999)
6. Van de Velde, M.: Signal Validation in Electroencephalography Research. PhD thesis, Technische Universiteit Eindhoven, Netherlands (January 2000)
7. Woon, W.L., Cichocki, A., Vialatte, F., Musha, T.: Techniques for early detection of Alzheimer's disease using spontaneous EEG recordings. Physiological Measurement 28(4), 335–347 (2007)
8. Melissant, C., Ypma, A., Frietman, E.E., Stam, C.J.: A method for detection of Alzheimer's disease using ICA-enhanced EEG measurements. Artif. Intell. 33(3), 209–222 (2005)
9. Escudero, J., Hornero, R., Poza, J., Abásolo, D., Fernández, A.: Assessment of classification improvement in patients with Alzheimer's disease based on magnetoencephalogram blind source separation. Artif. Intell. Med. 43, 75–85 (2008)
10. Cichocki, A., Amari, S., Siwek, K., Tanaka, T., Phan, A.-H.: ICALAB version 3 toolbox, http://www.bsp.brain.riken.jp/ICALAB/
11. Henderson, G., Ifeachor, E., Hudson, N., Goh, C., Outram, N., Wimalaratna, S., Del Percio, C., Vecchio, F.: Development and assessment of methods for detecting dementia using the human electroencephalogram. IEEE Trans. Biom. Eng. 53, 1557–1568 (2006)
12. Barlow, J.S.: The Electroencephalogram: Its Patterns and Origins. MIT Press, Cambridge (1993)
13. Tichavský, P., Yeredor, A., Nielsen, J.: A Fast Approximate Joint Diagonalization Algorithm Using a Criterion with a Block Diagonal Weight Matrix. In: Proc. ICASSP 2008, Las Vegas, USA (2008)
14. Yeredor, A.: Blind separation of Gaussian sources via second-order statistics with asymptotically optimal weighting. IEEE Signal Processing Letters 7, 197–200 (2000)
15. Tichavský, P., Doron, E., Yeredor, A., Nielsen, J.: A Computationally Affordable Implementation an Asymptotically Optimal BSS algorithm for AR Sources. In: EUSIPCO 2006, Florence, Italy (2006)

Global Minimization of the Projective Nonnegative Matrix Factorization

Zhijian Yuan

Neural Networks Research Centre
Helsinki University of Technology
P.O. Box 5400, 02015 HUT, Finland
zyuan@cis.hut.fi

Abstract. The Nonnegative Matrix Factorization (NMF) is a widely used method in approximating high dimensional data. All the NMF type methods find only the local minimizers. In this paper, we use filled function method to find the global minimizer of the Projective Nonnegative Matrix Factorization optimal problem.

Keywords: Projective Nonnegative Matrix Factorization, global optimization, filled function.

1 Introduction

Non-negative Matrix Factorization (NMF) by Lee and Seung [1] has been a very useful technique in approximating high dimensional data where the data are comprised of non-negative components. A large number of algorithms have been developed using NMF method. From the ideas of Singular Value Decomposition (SVD) and NMF, we have proposed the Projective Non-negative Matrix Factorization (P-NMF) [7], for learning spatially localized, parts-based representations of visual patterns. For the given nonnegative $m \times n$ matrix $\mathbf{V}$, find the nonnegative $m \times r$ matrix $\mathbf{W}$ $(r << m, n)$, such that it solves the following optimal problem

$$\min_{\mathbf{W} \geq 0} ||\mathbf{V} - \mathbf{W}\mathbf{W}^T\mathbf{V}||. \tag{1}$$

All the NMF type methods find the local minimizers. Althogh many mathematical methods have been proposed to search for a globally optimal solution of a given function, the computational problem due to the high dimention of the data set is the main issue. In this paper, we will solve the global minimizer of the equation (1) using the filled function method.

2 The Filled Function

The idea of using filled functions to solve the unconstrained global minimization of a continuous function $F(x)$, $x \in R^n$ comes from R.P. Ge([2] [3] [4]). Let $\mathbf{X}$ be a

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 987–993, 2009.

closed and bounded nonempty set which contains a finite number of minimizers of the function $F(x)$, and $x_k^* \in \mathbf{X}$ be a known local minimizer of $F(x)$ with $F(x_k^*) > F^* = min\{F(x)|x \in X\}$. The basic idea of the filled function methods is to construct an auxiliary function, called filled function of $F(x)$, such that minimizing the filled function will generate a point x_{k+1} in a basin (a particular connected domain around a local minimizer, see Definition below) of F(x) lower than the basin B_k of $F(x)$ at x_k^*. Then the minimization of the function $F(x)$ can be restarted at the point x_{k+1} to generate a new minimizer x_{k+1}^* of $F(x)$ with $F(x_{k+1}^*) < F(x_k^*)$. Repeat the process until a global minimizer of $F(x)$ is found. The filled function is updated at successively local minimizers of $F(x)$. The filled function at a local minimizer x_k^* of $F(x)$ is required to reach its maximum at x_k^*, to have neither a minimizer nor a saddle point in the basin B_k^* and in any basin of $F(x)$ higher than B_k^*, and to have minimizers or saddle points in basins of $F(x)$ lower than B_k^*.

2.1 Essentials of the Filled Function

The filled function method is concerned with finding the global minimizer of a multi-variable function f on R^n, under the following assumptions

1. f is continuously differentiable,
2. f has only a finite number of minimizers,
3. $f(x) \to \infty$ as $||x|| \to \infty$.

These assumptions imply that there exists a closed bounded domain Ω, called operating region, and it contains all the minima of $f(x)$.

To analyze the filled function method, we need the following concepts [2]:

Definition 1. *A basin of $f(x)$ at an isolated minimizer x_1 is a connected domain $\mathbf{B}_1$ which contains x_1 and in which starting from any point the steepest descent trajectory of $f(x)$ converges to x_1, but outside which the steepest descent trajectory of $f(x)$ does not converge to x_1.*

Definition 2. *A hill of $f(x)$ at x_1 is the basin of $-f(x)$ at its minimizer x_1, if x_1 is a maximizer of $f(x)$.*

Definition 3. *A local minimizer $\mathbf{X}_2$ is said to be higher than x_1 if and only if $f(x_2) > f(x_1)$, and for this case, $\mathbf{B}_2$ is said to be higher basin than $\mathbf{B}_1$.*

Definition 4. *A function $F(x)$ is called a filled function of $f(x)$ at x_1 if*

(1) x_1 is a maximizer of $F(x)$ and the whole basin $\mathbf{B}_1$ becomes a part of a hill of $F(x)$;

(2) $F(x)$ has no stationary points in any $\mathbf{B}_h s$;

(3) There is a point x' in a $\mathbf{B}_l$ (if such a basin exists) that minimizes $F(x)$ on the line through x and x_1

In the context of numerical optimization, the filled function method consists of a two-phase iterative approach called the sequential alternated minimization technique:

1. Local minimization: In this phase, a local minimizer x_1 of $f(x)$ is found. Any available local minimization method such as NMF, PNMF or the quasi-Newton method can be used to find a minimizer.
2. Filling: In this phase, a filled function $F(x)$ is constructed at the point x_1, and the minimization procedure is applied to the filled function. $F(x)$ has no stationary points in any higher basins B_h and does have a minimizer point in a lower basin B_l (if it exists). Then minimization of $F(x)$ starts from a point near x_1. Phase 2 ends when such an x_s is found that x_s is in a B_l. Then reenter phase 1, with x_s as the starting point, to find a new local minimizer x_2 of $f(x)$ (if such one exists), and so on.

The above process is repeated until the global minimizer is found.

Several popular filled functions have been proposed [2] [3] [4]:

$$P(x, r, \rho) = \exp(-||x - x_1||^2/\rho^2)/(r + f(x)) \tag{2}$$

$$G(x, r, \rho) = -\{\rho^2 \ln(r + f(x)) + ||x - x_1||^p\} \tag{3}$$

$$Q(x, a) = -[f(x) - f(x_1)] \exp(a||x - x_1||^p) \tag{4}$$

where $p = 1, 2$. r and ρ are the adjustable parameters, and a is an adjustable positive weight factor.

2.2 Properties of the Filled Function

Consider the optimal problem (1), define the function

$$f(\mathbf{W}) = ||\mathbf{V} - \mathbf{W}\mathbf{W}^T\mathbf{V}||^2. \tag{5}$$

In the paper, we will use the one parameter filled function which is defined as [6]:

$$F(\mathbf{W}) = -\frac{1}{(f(\mathbf{W}) - f(\mathbf{W}_0))^{1/2}} - a||\mathbf{W} - \mathbf{W}_0||^2, \tag{6}$$

where a is a positive real weight factor, $\mathbf{W}_0$ is a local minimizer of the function $f(\mathbf{W})$.

Following the definition of $F(\mathbf{W})$, it is easy to see that the local minimizers of $f(\mathbf{W})$ are the local maximizers of $F(\mathbf{W})$. We also have

Theorem 1. *Given direction $d \in R^{mr}$ and $f(\mathbf{W}) > f(\mathbf{W}_0)$, if $\Sigma_{ij} d_{ij} (\bigtriangledown f(\mathbf{W}))_{ij} \geq 0$ and $\Sigma_{ij} d_{ij} (\mathbf{W} - \mathbf{W}_0)_{ij} > 0$, or $\Sigma_{ij} d_{ij} (\bigtriangledown f(\mathbf{W}))_{ij} > 0$ and $\Sigma_{ij} d_{ij} (\mathbf{W} - \mathbf{W}_0)_{ij} \geq 0$, then d is a descent direction of $F(\mathbf{W})$ at point $\mathbf{W}$.*

Define

$$a_l(\mathbf{W}) = -\frac{\Sigma_{ij} d_{ij} (\bigtriangledown f(\mathbf{W}))_{ij}}{4(f(\mathbf{W}) - f(\mathbf{W}_0))^{3/2} \Sigma_{ij} d_{ij} (\mathbf{W} - \mathbf{W}_0)_{ij}} \tag{7}$$

We have

Theorem 2. *If $f(\mathbf{W}) > f(\mathbf{W}_0)$ and $\Sigma_{ij} d_{ij} (\bigtriangledown f(\mathbf{W}))_{ij} < 0$ and $\Sigma_{ij} d_{ij} (\mathbf{W} - \mathbf{W}_0)_{ij} > 0$, then the sign of $\Sigma_{ij} d_{ij} (\bigtriangledown F(\mathbf{W}))_{ij}$ is that of $a_l(\mathbf{W}) - a$.*

These results clearly characterize the filling property of the the function $F(\mathbf{W})$. Particularly, in the ascent region of the current basin B_1 or a higher basin than B_1, d is always a descent direction of $F(\mathbf{W})$. On the other hand, in the descent region of a higher basin than B_1, d is still a descent direction of $F(\mathbf{W})$ provided that the weight factor a is sufficiently large. Furthermore, in a lower bassin than B_1, d may become an ascent direction of $F(\mathbf{W})$ and this possibility does exists. Therefore, $F(\mathbf{W})$ must have a stationary point along d.

The behavior of the filled function may be well interpreted graphically for one variable function $f(x)$, see Figure 1. Function $f(x)$ has three basins, $B_1 = (0.5, 2)$, a higher basin $B_H = (2, 4)$, and a lower basin $B_L = (4, 6)$. The filled function starts from the basin B_1, and descends along the positive x-axis in this basin. It continues decreasing in the higher basin B_H for a sufficient large a. When it arrive a lower basin B_L, it has a stationary point, which is a local minimizer of the filled function. From this stationary point, minimizing the function $f(x)$ will reach a lower minimizer of the function $f(x)$.

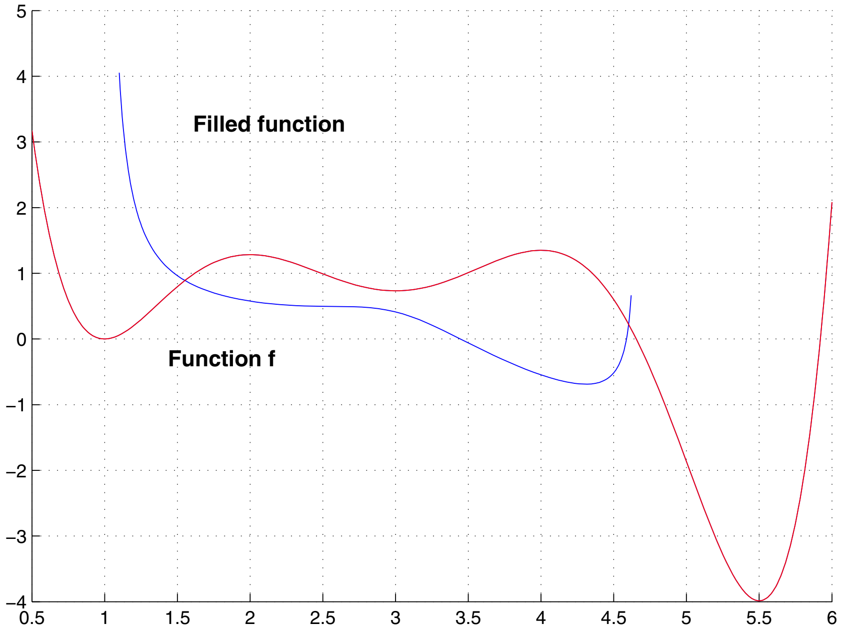


Fig. 1. Plot of the function and the related filled function

The weight factor a plays a crucial role in a filled function. Theoretically, the value of a must be sufficietly large to preserve a desirable filling capability. Computationally, the value of a should be small to make the numerical procedures healthy, because there is a product of a and the norm in the formation. Therefore, a filled function is a robust one if the value of $a_l(\mathbf{W})$ is small for the given particular $\mathbf{W}$.

3 Algorithm

To use the filled function, we first need to prove that all the minimizers of the function are isolated.

Theorem 3. *The minimizers of the function* $f(\mathbf{W}) = ||V - \mathbf{W}\mathbf{W}^T\mathbf{V}||^2$ *are isolated.*

Proof. If there exists one point $\mathbf{W}$ that is not isolated minimizer, that is, there is a series of nonnegative matrices $\mathbf{W}_n \neq \mathbf{W}$, $\mathbf{W}_n \approx \mathbf{W}$ as $n \to \infty$, such that $f(\mathbf{X}) = f(\mathbf{W}_n)$. Since the minima $\mathbf{W}$, $\mathbf{W}_n$ of the optimal problem 1 satisfies

$$trace(\mathbf{W}\mathbf{W}^T\mathbf{V}\mathbf{V}^T) = trace(\mathbf{W}\mathbf{W}^T\mathbf{V}\mathbf{V}^T\mathbf{W}\mathbf{W}^T). \tag{8}$$

$$trace(\mathbf{W}_n\mathbf{W}_n^T\mathbf{V}\mathbf{V}^T) = trace(\mathbf{W}_n\mathbf{W}_n^T\mathbf{V}\mathbf{V}^T\mathbf{W}_n\mathbf{W}_n^T). \tag{9}$$

Let $\Delta\mathbf{W}_n = \mathbf{W}_n - \mathbf{W}$, we have

$$\begin{aligned} 0 &= f(\mathbf{X}) - f(\mathbf{W}_n) \\ &= 2tr(\Delta\mathbf{W}_n\mathbf{W}^T\mathbf{V}\mathbf{V}^T) + tr(\Delta\mathbf{W}_n\Delta\mathbf{W}_n^T\mathbf{V}\mathbf{V}^T) \end{aligned}$$

holds for all n. Since $\Delta\mathbf{W}_n \to 0$ as $n \to \infty$ and $\Delta\mathbf{W}_n \neq 0$, so for the second term of the above equation, we have $tr(\Delta\mathbf{W}_n\Delta\mathbf{W}_n^T\mathbf{V}\mathbf{V}^T) = O(||\Delta\mathbf{W}_n||^2)$ if $tr(\Delta\mathbf{W}_n\Delta\mathbf{W}_n^T\mathbf{V}\mathbf{V}^T) \neq 0$ for n large enough. However, $2tr(\Delta\mathbf{W}_n\mathbf{W}^T\mathbf{V}\mathbf{V}^T) = O(||\Delta\mathbf{W}_n||)$, their sum cannot be equal to zero. This is not true from the above equation. Therefore, we conclude that $tr(\Delta\mathbf{W}_n\Delta\mathbf{W}_n^T\mathbf{V}\mathbf{V}^T) = 0$ as $n \to \infty$. This means that $||\Delta\mathbf{W}_n^T\mathbf{V}||_F = 0$, with the assumption that $\mathbf{V}$ is a full rank matrix, we have $\Delta\mathbf{W}_n = 0$ for n large enough. Thus $\mathbf{W}$ is isolated, and the theorem is proved.

Now we can use the filled function to calculate the global minimizer of the function $f(\mathbf{W})$. According to the properties of the filled function, we have the following algorithm to solve the optimal problem (1)

1. Initialization. Specify $\mathbf{W}_0$, a, f_{min}.
2. Finfing local minimizer of $f(\mathbf{W})$. Starting from $\mathbf{W}_0$, find the local minimizer $\mathbf{W}_1$. This can be done by, for example, PNMF algorithm. If $f(\mathbf{W}_1) \leq f_{min}$, then $f(\mathbf{W}_1) \to f_{min}$; otherwise, $2a \to a$.
3. Find the minimizer of filled function. Starting from $\mathbf{W}_1$, activate the linear search procedure to minimize the filled function, arrive at point $\mathbf{W}$. If $f(\mathbf{W}) < f(\mathbf{W}_1)$ or $\mathbf{W}$ is the minimizer of the filled function, then $\mathbf{W} \to \mathbf{W}_0$, goto step 2.
4. Stop when the global minimum found.

4 Simulation

4.1 Vector Case and Low Dimention Case

In the vector case, the minima $\mathbf{W}$ of the optimal problem 1 satisfies

$$trace(\mathbf{W}\mathbf{W}^T\mathbf{V}\mathbf{V}^T) = trace(\mathbf{W}\mathbf{W}^T\mathbf{V}\mathbf{V}^T\mathbf{W}\mathbf{W}^T). \tag{10}$$

If $\mathbf{W}$ is a vector, then the minima of the optimal problem (1) is equivalent to the maximizer of the following optimal problem

$$\max_{\mathbf{W}\geq 0, ||\mathbf{W}||=1} trace(\mathbf{W}\mathbf{W}^T\mathbf{V}\mathbf{V}^T) \tag{11}$$

It is easy to see that the function $trace(\mathbf{W}\mathbf{W}^T\mathbf{V}\mathbf{V}^T)$ is a polynomial with power 2 and nonnegative coefficients. It is a convex function in the domain $0 \leq ||\mathbf{W}|| \leq 1$. The local optimal solutions of the above optimal problem are also the global solutions.

As an example, consider a simple case, the vector $\mathbf{W} = (w_1, w_2)^T$, the function $f(\mathbf{W})$ as

$$f(\mathbf{W}) = ||\mathbf{V} - \mathbf{W}\mathbf{W}^T\mathbf{V}||^2 \tag{12}$$

where $\mathbf{V}$ is a 2×3 matrix.

Running both PNMF and global algorithms obtained the same optimall point.

Figure 2 shows the function $f(\mathbf{W})$.

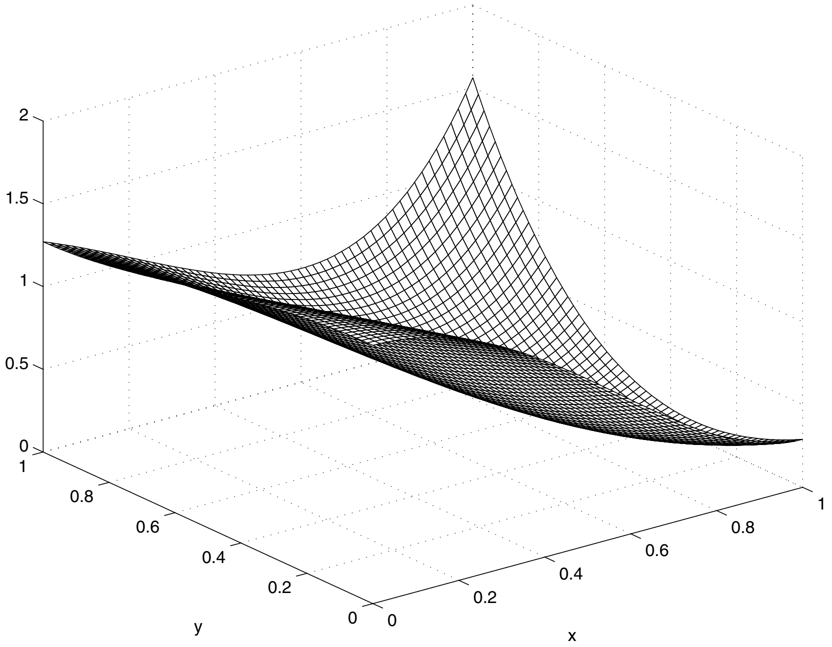


Fig. 2. Plot of the function $f(\mathbf{W})$

For the low dimention case, we have tested some functions. For example, $\mathbf{V}$ is a 3×3 identity matrix, $\mathbf{W}$ is a 3×2 matrix, the function f is

$$\begin{aligned} f(\mathbf{W}) = &\sum_{ij} \mathbf{w}_{ij}^4 + 2\mathbf{w}_{11}^2\mathbf{w}_{12}^2 + 2\mathbf{w}_{11}^2\mathbf{w}_{21}^2 + 2\mathbf{w}_{11}^2\mathbf{w}_{31}^2 + 2\mathbf{w}_{21}^2\mathbf{w}_{22}^2 + 2\mathbf{w}_{21}^2\mathbf{w}_{31}^2 \\ &+ 2\mathbf{w}_{12}^2\mathbf{w}_{22}^2 + 2\mathbf{w}_{12}^2\mathbf{w}_{32}^2 + 2\mathbf{w}_{22}^2\mathbf{w}_{32}^2 + 2\mathbf{w}_{31}^2\mathbf{w}_{32}^2 + 4\mathbf{w}_{11}\mathbf{w}_{21}\mathbf{w}_{12}\mathbf{w}_{22} \\ &+ 4\mathbf{w}_{11}\mathbf{w}_{31}\mathbf{w}_{12}\mathbf{w}_{32} + 4\mathbf{w}_{21}\mathbf{w}_{31}\mathbf{w}_{22}\mathbf{w}_{32} - 2\sum_{ij} \mathbf{w}_{ij}^2 + 3. \end{aligned}$$

The values of the function f by P-NMF and the filled function method are 1.0019 and 1.0000, respectively.

4.2 Image Data

We used face images from the MIT-CBCL database as experimental data. The training data set contains 2429 faces. Each face has $19 \times 19 = 361$ pixels and has been histogram-equalized and normalized so that all pixel values are between 0 and 1. Thus the data matrix $\mathbf{V}$ which now has the faces as columns is 361×2429. This matrix was compressed to rank $r = 49$ using P-NMF expansions and the filled function method.

The P-NMF gives a local minimizer $\mathbf{W}$, the value of function f at $\mathbf{W}$ is $2.3768e + 08$. The filled function method gives the global minimization value $2.3460e + 08$.

5 Conclusion

In this paper, the filled function method is used to find the global minimization of the PNMF optimal problem. The experiments show that the filled function method works and it is computable. For the original NMF problem, we can also find the global minimizer similarly using filled function method.

References

1. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature 401, 788–791 (1999)
2. Ge, R.P.: A filled function method for finding a global minimizer of a function of several variables. Math. Programming 46, 191–204 (1990)
3. Ge, R.P., Qin, Y.F.: A class of filled functions for finding a global minimizer of a function of several variables. J. Optim. Theory Appl. 54(2), 241–252 (1987)
4. Ge, R.P., Qin, Y.F.: The global convexized filled functions for globally optimization. Appl. Math. Comput. 54(2), 131–158 (1990)
5. Liu, X.: A class of generalized filled functions with improved computability. J. Comput. Appl. Math. 137(1), 62–69 (2001)
6. Liu, X.: A computable filled function used for global minimization. Applied Mathematics and Computation 126, 271–278 (2002)
7. Yuan, Z., Oja, E.: Projective nonnegative matrix factorization for image compression and feature extraction. In: Kalviainen, H., Parkkinen, J., Kaarna, A. (eds.) SCIA 2005. LNCS, vol. 3540, pp. 333–342. Springer, Heidelberg (2005)

Learning Sparse Representations Using a Parametric Cauchy Density*

Ling-Zhi Liao

School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, 210044, China
sophiallz@163.com

Abstract. For extracting sparse structures in images adaptively, the prior probabilities over the coefficients are modeled with a flexible parametric Cauchy density, which can describe a class of super-Gaussian distributions by varying the steepness and the scale parameters in the density function. The derivatives of the sparseness cost function are continuous at each point of its domain, which is convenient for gradient techniques based learning algorithms, and may provide a better approximation of the volume contribution from the prior. The performance of the flexible prior is demonstrated on a set of natural images.

1 Introduction

Previous work [1,2] using sparsity criteria have suggested that the basis functions may best encode natural images in terms of sparse, independent components, for such basis functions are comparable with the receptive fields of simple cells in mammalian primary visual cortex, which are spatially localized, oriented, and band-pass in spatial frequency [3-5]. Unlike the ICA [6,7] techniques, the sparse coding strategies can be combined with overcomplete representations [8-10], which allows for a better approximation of the underlying statistical distribution of the data. The research on sparse coding is hereby helpful for both vision research and image processing.

Essentially, the probabilistic inference of sparse coding is according to Bayes' theorem [11]. The main advantage of such a probabilistic decision is that it is allowed to incorporate prior knowledge into the data analysis, in order to bias the interpretation of the data in the direction of expectation. However, the coefficient prior distributions, $P(s_i)$, in most of the previous sparse coding models [1,8,9,2] are assumed to be fixed, not adapted to the data being analyzed, which may lead to an inaccurate prior assumption.

It has been observed that the sparse distributions over wavelet coefficients of images can be well modeled with a generalized Laplacian density (GLD) [12], or called generalized Gaussian density (GGD) [13]. Thus one possibility for improving the prior in sparse coding is to model $P(s_i)$ with a GGD function. By varying the shape parameter in the GGD model, a wide class of statistical distributions can be

* Supported by the Science Foundation of Nanjing University of Information Science and Technology.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 994–1002, 2009.

characterized, including Gaussian, Laplacian, and some other super- and sub-Gaussian densities. However, when the shape parameter is equal to or less than the value of one, the derivative of the sparseness cost function, or $\log P(s_i)$, at zero is not continuous, which will be troublesome for gradient techniques based learning algorithms for finding the most probable coefficients [9].

In this work, the prior distributions $P(s_i)$ are characterized by a parametric Cauchy density (PCD) function. By varying the steepness and the scale parameters in the PCD model, a class of super-Gaussians can be characterized. Such a sparseness property makes the PCD model suitable for the sparse coding framework, and makes the priors adaptive to the data being analyzed. Moreover, compared with the GGD model, the derivatives of $\log P(s_i)$ in the PCD model are continuous everywhere in its domain. Such a smoothness property may provide a better approximation of the volume contribution from the prior, and allows for gradient descent solutions when seeking to maximize the posterior distribution over the coefficients.

The rest of the paper is organized as follows: Section 2 shows two properties of the proposed PCD model, including *sparseness* and *smoothness*; and gives a simple method to estimate the values of the steepness and the scale parameters. In Section 3, we inference the gradient learning rules for the coefficients and the basis functions, when applying the PCD model into the sparse coding framework. Section 4 reports experiments carried out on a set of natural images and the results. Section 5 presents the conclusions.

2 Parametric Cauchy Density

Before discussing the PCD model, we firstly introduce the GGD model briefly. A GGD function with zero mean is usually given by [13]

$$P(x) = W \exp\left[-\left(|x|/\theta\right)^q\right] \tag{1}$$

The logarithm is

$$\log P(x) = \log W - \left(|x|/\theta\right)^q \tag{2}$$

where x is the random variable; the parameters $q > 0$ and $\theta > 0$ are called the shape parameter and the scale parameter, respectively; W is a normalization constant to ensure that $\int P(x)dx = 1$.

The derivative function $d(x) = \partial \log P(x)/\partial x$ is

$$d(x) = -|x|^{q-1}\operatorname{sgn}(x)q/\theta^q \propto -|x|^{q-1}\operatorname{sgn}(x) \tag{3}$$

where $\operatorname{sgn}(x)$ is the sign function of x. Especially, when $P(x)$ is a Gaussian density ($q = 2$), we have $d(x) \propto -x$; when $P(x)$ is as sparse as a near-delta function ($q \to 0$), we have $d(x) \propto -x^{-1}$.

2.1 Sparseness

The PCD function, in this work, is given by

$$P(x) = C\exp\left[-\beta\log\left(1+(x/\alpha)^2\right)\right] \tag{4}$$

The logarithm is

$$\log P(x) = \log C - \beta\log\left(1+(x/\alpha)^2\right) \tag{5}$$

in which the parameters α and β are both larger than zero; and $C = \Gamma(\beta)\big/(\alpha\sqrt{\pi}\Gamma(\beta-1/2))$ is used to normalizing the density function, where $\Gamma(\cdot)$ is a Gamma function. It is obvious that the PCD model contains the standard Cauchy distribution as a special case, when $\alpha = 1$ and $\beta = 1$.

When the density $P(x)$ is characterized by the PCD model, the derivative function is

$$d(x) = -2\beta\frac{x/\alpha^2}{1+(x/\alpha)^2} \tag{6}$$

If $\alpha \gg x$ or $x/\alpha \ll 1$, we will have

$$d(x) \approx -2\beta\, x/a^2 \propto -x \tag{7}$$

which indicates that the PCD model is close to a Gaussian. Meanwhile, if $\alpha \ll x$ or $x/\alpha \gg 1$, we will have

$$d(x) \approx -2\beta x^{-1} \propto -x^{-1} \tag{8}$$

which suggests that the PCD model becomes a near-delta function at zero. Thus it can be seen that the PCD model can describe a class of statistical distributions including Gaussian and super-Gaussian densities by varying the values of α. Hereby we named α the steepness parameter, which mainly controls the decreasing rate of the PCD peak. The parameter β is mainly relative to the variance of the random variable, therefore referred to as the scale parameter.

2.2 Smoothness

Obviously, for a GGD function with $q \le 1$, the derivative function $d(x)$ has a discontinuity at $x = 0$. When the distribution $P(x)$ is characterized by the proposed PCD model, however, the derivative function $d(x)$ are continuous at each point of its domain, for

$$\lim_{x\to a} d(x) = d(a) \tag{9}$$

where a is an any point in the domain of d.

In other words, the derivative function $d(x)$ is a smooth function, when the distribution $P(x)$ is characterized by a PCD model. Furthermore, when the prior distributions over sparse coefficients are modeled with the PCD function, the derivatives of the sparseness cost function are continuous at each point of its domain, which may provide a better approximation of the volume contribution from the prior [9], and allows for gradient descent solutions when seeking for the most probable coefficients.

2.3 Estimating α and β

Given the observations $x=\left[x_1,\cdots,x_n\right]$, estimating the values of α and β is accomplished by finding the values that maximize the log-likelihood function

$$L(\alpha,\beta)=\log\left(\prod_{i=1}^{n} C\exp\left[-\beta\log\left(1+\left(x_i/\alpha\right)^2\right)\right]\right) \tag{10}$$

The gradients in the directions of α and β are

$$\Delta\alpha=\frac{\partial L(\alpha,\beta)}{\partial\alpha}=\frac{n}{\alpha}\left(2\beta-1\right)-\beta\sum_{i=1}^{n}\frac{2\alpha}{\alpha^2+x_i^2} \tag{11}$$

$$\Delta\beta=\frac{\partial L(\alpha,\beta)}{\partial\beta}=n\left[\Psi(\beta)-\Psi(\beta-1/2)\right]-\sum_{i=1}^{n}\left[\log\left(\alpha^2+x_i^2\right)-2\log\alpha\right] \tag{12}$$

where $\Psi(\cdot)$ is the derivate of the Gamma function $\Gamma(\cdot)$.

Then the optimal values for α and β can be obtained by employing the gradient techniques based learning algorithms. Fig. 1 shows an example of a histogram together with the fitted PCD using the ML estimators. The histogram is of the coefficient samples for a learned basis function with the sparse coding strategy in [8]. The estimated parameters are: $\alpha=0.2701$ and $\beta=2.0395$. The fits are generally good. As a result, with the parameters α and β for the PCD model, we can accurately capture the distributions of the sparse basis coefficients.

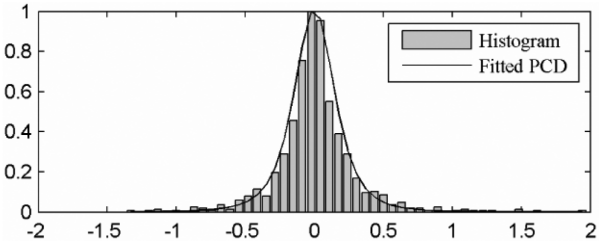


Fig. 1. Sparse basis coefficient histogram fitted with a PCD

3 Application to Sparse Coding

3.1 Principles of Sparse Coding

The basic assumption of sparse coding is that an image can be represented by a linear superposition of basis functions plus noise [8,2]

$$\mathbf{I} = \mathbf{A}\mathbf{s} + \mathbf{N} \tag{13}$$

where $\mathbf{I} = [I_1, \cdots, I_j, \cdots I_L]^{\mathrm{T}}$ is an L- element input image, $\mathbf{A}$ is an $L \times M$ matrix whose columns $A_1, \cdots, A_i, \cdots A_M$ are the basis functions, $\mathbf{s} = [s_1, \cdots, s_i, \cdots s_M]^{\mathrm{T}}$ is a M-element coefficient vector, and $\mathbf{N}$ represents a noise sampled from a Gaussian distribution. The two-fold goals of sparse coding are to find a good matrix, $\mathbf{A}$, for coding the entire input images, and to infer the proper coefficients, $\mathbf{s}$, for each individual image.

Based on the Bayes' theorem, one probabilistic approach to inferring the coefficients is to find a coefficient vector, $\hat{\mathbf{s}}$, such that

$$\hat{\mathbf{s}} = \underset{\mathbf{s}}{\arg\max}\left[\log P\left(\mathbf{I} \mid \mathbf{A}, \mathbf{s}\right) + \log P\left(\mathbf{s}\right)\right] = \underset{\mathbf{s}}{\arg\min}\left[\frac{\left|\mathbf{I} - \mathbf{A}\mathbf{s}\right|^2}{2\sigma_N^2} - \sum_{i=1}^{M} \log P(s_i)\right] \tag{14}$$

Here

$$P(\mathbf{I} \mid \mathbf{A}, \mathbf{s}) \propto \exp\left(-\left|\mathbf{I} - \mathbf{A}\mathbf{s}\right|^2 \big/ 2\sigma_N^2\right) \tag{15}$$

is the probability of the image under the generative model given the coefficients, in which σ_N is the standard variance of the additive noise. Meanwhile,

$$P(\mathbf{s}) = \prod_i P(s_i) \tag{16}$$

is the prior distribution over the coefficients. Obviously the concept of statistical independence is incorporated into the image model since $P(\mathbf{s})$ is a factorial distribution. The concept of sparseness is also incorporated into the model by assuming that the probability distribution of each coefficient, $P(s_i)$, usually has a sparse shape.

The basis functions are then adapted by maximizing the average log-likelihood of the images at the posterior maximum [8,2]. In another word, the goal is to find a set of basis functions, $\hat{\mathbf{A}}$, such that

$$\hat{\mathbf{A}} = \underset{\mathbf{A}}{\arg\max}\left\langle \log P\left(\mathbf{I} \mid \mathbf{A}, \mathbf{s} = \hat{\mathbf{s}}\right)\right\rangle = \underset{\mathbf{s}}{\arg\min}\left\langle \frac{\left|\mathbf{I} - \mathbf{A}\hat{\mathbf{s}}\right|^2}{2\sigma_N^2} \right\rangle \tag{17}$$

where the brackets $\langle \ \rangle$ mean "averaged over all images".

3.2 Improving Prior

In this work, the prior distributions $P\left(s_i\right)$ are characterized by the proposed PCD model and in the form of

$$P(s_i | \alpha_i, \beta_i) = C \exp\left[-\beta_i \log\left(1 + (s_i/\alpha_i)^2\right)\right] \tag{18}$$

Along with the adaptation of the basis functions, the values of α_i and β_i for each s_i can be updated during learning. Thus the priors can be adaptive to the input data.

Suppose s_{ik} be the response of i-th basis function to the k-th input image. Then the vector $[s_{i1}, \cdots, s_{ik}, \cdots, s_{in}]$ can be regarded as a set of observations for the i-th coefficient variable, s_i. Once the observations for s_i are obtained, the values of α_i and β_i can be estimated by employing the method mentioned in section 2.3. Note that the values of α_i and β_i should be (re)estimated periodically during learning. It is not wise to adjust α_i and β_i every time when the basis functions are updated. In this work, the values of α_i and β_i are updated per 1000 updates of the basis functions.

The gradient descent learning rule for the coefficient s_i is then

$$\Delta s_i = -\frac{1}{\sigma_N^2} A_i^{\mathrm{T}} (\mathbf{I} - \mathbf{As}) + \frac{2\beta_i s_i}{s_i^2 + \alpha_i^2} \tag{19}$$

The learning rule for the basis functions via the natural gradient algorithm [14] is

$$\Delta \mathbf{A} = -\frac{\mathbf{AA}^{\mathrm{T}}}{\sigma_N^2} \left\langle (\mathbf{I} - \mathbf{A}\hat{\mathbf{s}})\hat{\mathbf{s}}^{\mathrm{T}} \right\rangle \tag{20}$$

Note that the basis functions must be rescaled [8,2] after each learning step in order to ensure that they do not grow without bound.

In summary, the whole processing of learning can be concluded as follows:

Step 1: Let $t = 0$; For each $i \in [1, M]$, let $\alpha_i = a$, $\beta_i = b$, where a and b are the initial values of α_i and β_i, respectively; and let $\mathbf{A}$ be a random $L \times M$ matrix.

Step 2: For each $i \in [1, M]$, infer the optimal coefficient $\hat{s}_i$ given $\mathbf{A}$, α_i and β_i, based on eq. (19).

Step 3: Update $\mathbf{A}$ based on eq. (20).

Step 4: Let $t = t + 1$. If $t = 1000m$ ($m = 1, 2, 3, \cdots$), then go to *Step* 5. Otherwise, go to *Step* 2.

Step 5: Compute the observations $[s_{i1}, \cdots, s_{ik}, \cdots, s_{in}]$ for each s_i, and then update the values of α_i and β_i.

Step 6: Stop or go to *Step* 2.

4 Experiments and Results

The images for training are ten 512×512 whitened images in the dataset of [8]. The basis functions were trained on two data sets, which were obtained by extracting 8×8 and 12×12 image patches randomly from the ten whitened images, respectively. Note that the patches were repeatedly resampled throughout training to avoid reuse of any one set of patches. Meanwhile, the basis matrix $\mathbf{A}$ was adapted using eq.(20) on blocks of 200 patches. Here we learned complete and $2\times$'s overcomplete representations of the two data sets. For both data sets, the initial values a and b were 0.316 and 2.2, respectively; the initial basis functions were generated by setting the basis elements to random values between [-0.5, 0.5], and scaling each basis function to have one variance. In addition, the standard variance σ_N was set to be 0.01. And the learning stopped when $t = 10000$. Namely, the values of α_i and β_i were updated ten times during learning. Every time when we updated the values of α_i and β_i, we firstly extracted 2000 patches at random from the ten whitened images (200 patches per image), and then computed the observations vector $\left[s_{i1}, \cdots, s_{ik}, \cdots, s_{in}\right]$ ($n = 2000$) for each s_i using eq.(19). Given the observations, the values of α_i and β_i could be estimated and updated.

Fig. 2 shows the learned basis functions training from the 12×12 data set when the code was $2\times$'s overcomplete. Obviously, like the Gabor functions[15], the basis functions obtained with the proposed PCD prior model were also localized, oriented, and band-pass. Namely, the learned basis functions were comparable with the receptive fields of simple cells in mammalian primary visual cortex.

To make a comparable analysis, we also learned complete and $2\times$'s overcomplete representations on the two data sets when the prior was fixed. The imposed prior was chosen to be a Cauchy density in the form of eq.(18), too, but the values of α_i and β_i were not updated during learning, and fixed at 0.316 and 2.2 (as same as the initial values for the adaptive prior), respectively. The posterior distributions over four coefficients were shown in Fig. 3 (upper). The two left ones were from the complete and $2\times$'s overcomplete representations for the 8×8 data set, respectively; while two right ones were from the complete and $2\times$'s overcomplete representations for the 12×12 data set, respectively. The overlaid dashed line in each subplot was the imposed sparse prior. It was obvious that coefficients didn't fit the imposed prior very well. While Fig. 3 (lower) shows the posterior distributions over four coefficients from the complete and $2\times$'s overcomplete sparse representations for the two data sets, respectively, when the prior was adaptive to the data. The resulting priors learned from the data were overlaid (dashed line), too. The posteriors closely matched the priors, indicating that the proposed model may be a reasonable fit to the image data.

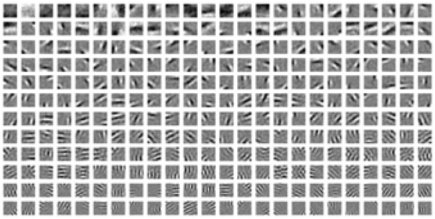


Fig. 2. Learned basis functions from training $2\times$'s overcomplete basis functions on the 12×12 data set

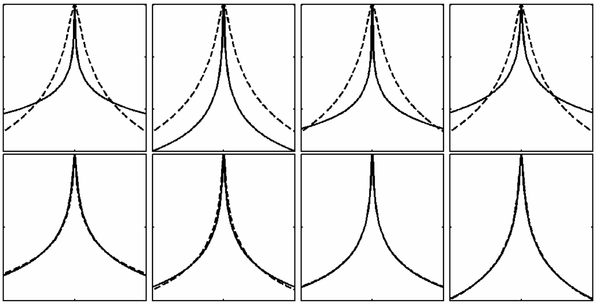


Fig. 3. Posterior distributions (solid lines) of coefficients with corresponding priors (dash lines). The priors in the upper graphs were fixed while the priors in the lower were adaptive to the input data. The y-axis in each subplot was plotted on a log scale.

Since our image model is in the probabilistic framework, it is not difficult to evaluate the coding cost of the representation. The estimated coding costs in Table 1 were calculated using the entropy method described in [9], the nature of which was to quantize the coefficients to a noise level and then calculate the total coefficient entropy. For each representation from Table. 1, the noise level for quantization was set to 0.01, as same as that used during the learning.

Table 1. Estimated coding efficiencies (bits per pixel) for different representations

			Adaptive prior	Fixed prior
Dataset	8×8	1× 's	3.7122 ± 0.4017	4.0229 ± 0.3846
		2× 's	3.0116 ± 0.2894	3.5373 ± 0.3302
	12×12	1× 's	3.7476 ± 0.3423	4.4093 ± 0.3126
		2× 's	3.2270 ± 0.2392	3.6529 ± 0.2207

The table shows that the overcomplete image codes have lower coding cost, despite the fact that there are more coefficients to code. Moreover, the table suggests that the image model in eq.(13) with an adaptive PCD prior on the basis function coefficients gives lower coding cost than that with a fixed prior, which indicates that the adaptive prior can improve the coding efficiency.

References

1. Olshausen, B.A.: Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381(6583), 607–609 (1996)
2. Olshausen, B.A.: Principles of image representation in visual cortex. Vis. Neurosci., 1603–1615 (2003)
3. Movshon, J.A., Thompson, I.D., Tolhurst, D.J.: "Receptive field organization of complex cells in the cat's striate cortex". J. Physiol. 283(1), 79–99 (1978)
4. Field, D.J.: Relations between the statistics of natural images and the response properties of cortical cells. J. Opt. Soc. Am. A 4(12), 2379–2394 (1987)
5. Ringach, D.L.: Spatial structure and symmetry of simple-cell receptive fields in macaque primary visual cortex. J. Neurophysiol. 88(1), 455–463 (2002)

6. Comon, P.: Independent component analysis, a new concept? Signal Processing 36(3), 287–314 (1994)
7. Bell, A.J., Sejnowski, T.J.: The 'independent components' of natural scenes are edge filters. Vis. Res. 37(23), 3327–3338 (1997)
8. Olshausen, B.A., Field, D.J.: Sparse coding with an overcomplete basis set: A strategy employed by V1. Vis. Res. 37(23), 3311–3325 (1997)
9. Lewicki, M.S., Sejnowski, T.J.: Learning Overcomplete Representations. Neurocomput. 2(12), 337–365 (2000)
10. Doi, E., Lewicki, M.S.: Sparse coding of natural images using an overcomplete set of limited capacity units. In: Proc. NIPS 2005, Vancouver, Canada, vol. 17, pp. 377–384. MIT Press, Cambridge (2005)
11. Bromiley, P.A., et al.: Bayesian and non-Bayesian probabilistic models for medical image analysis. Image Vis. Comput. 21(10), 851–864 (2003)
12. Buccigrossi, R.W., Simoncelli, E.P.: Image compression via joint statistical characterization in the wavelet domain. IEEE Trans. Image Process. 8(12), 1688–1701 (1999)
13. Do, M.N., Vetterli, M.: Wavelet-based texture retrieval using generalized Gaussian density and Kullback-Leibler distance. IEEE Trans. Image Process. 11(2), 146–158 (2002)
14. Amari, S.: Natural gradient works efficiently in learning. Neural Comput. 10(2), 251–276 (1998)
15. Lewicki, M.S., Olshausen, B.A.: Probabilistic framework for the adaptation and comparison of image codes. J. Opt. Soc. Am. A 16(7), 1587–1601 (1999)

A One-Layer Recurrent Neural Network for Non-smooth Convex Optimization Subject to Linear Equality Constraints*

Qingshan Liu[1] and Jun Wang[2]

[1] School of Automation, Southeast University, Nanjing, Jiangsu, China
qsh.liu@gmail.com
[2] Department of Mechanical and Automation Engineering
The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong
jwang@mae.cuhk.edu.hk

Abstract. In this paper, a one-layer recurrent neural network is proposed for solving non-smooth convex optimization problems with linear equality constraints. Comparing with the existing neural networks, the proposed neural network has simpler architecture and the number of neurons is the same as that of decision variables in the optimization problems. The global convergence of the neural network can be guaranteed if the non-smooth objective function is convex. Simulation results are provided to show that the state trajectories of the neural network can converge to the optimal solutions of the non-smooth convex optimization problems and show the performance of the proposed neural network.

1 Introduction

Consider the following nonlinear programming (NP) problem:

$$\begin{array}{ll} \text{minimize} & f(x), \\ \text{subject to} & Ax = b, \end{array} \tag{1}$$

where $x \in \mathbb{R}^n$, $f(x) : \mathbb{R}^n \to \mathbb{R}$ is convex continuous function but not smooth (i.e., not continuously differentiable), $A \in \mathbb{R}^{m \times n}$ is a full row-rank matrix (i.e., $\text{rank}(A) = m$), and $b \in \mathbb{R}^m$.

Convex programming has many applications in scientific and engineering areas, such as signal and image processing, manufacturing, optimal control, and pattern recognition. Non-smooth optimization has been widely utilized to minimax problems, parameter estimation and support vector machine learning. Recurrent neural networks based on hardware implementation are effective for online solutions of convex programming problems [1,2,3,4,5,6,7,8,9,10]. In 1986, Tank and Hopfield [1] first proposed a neural network for solving linear programming problems, which motivated the development of neural networks for

* The work described in this paper was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project no. CUHK417608E).

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 1003–1010, 2009.

solving linear and nonlinear programming problems. Kennedy and Chua [2] presented a neural network for solving nonlinear programming problems by utilizing the finite penalty parameter method and the network is convergent to an approximate optimal solution, and this neural network has an implementation problem when the penalty parameter is very large. Zhang and Constantinides [11] proposed the Lagrangian network which has two-layer structure and can be utilized to solve some strictly convex programming problems. Wang and Xia [5] proposed a primal-dual neural network and it can be utilized to some convex quadratic programming problems. Recently, the projection neural networks were proposed for solving general nonlinear programming problems, and these neural networks have well convergence properties and can globally converge to an exact optimal solution for convex programming problems [12][13]. In [10][14], we proposed some one-layer recurrent neural networks for solving linear and quadratic programming problems. In [15][16], the non-smooth optimization was investigated. In [15], a neural network model was proposed for solving non-smooth convex optimization subject to bound constraints. In [16], a two-layer recurrent neural network was constructed for solving non-smooth convex optimization subject to linear equality and bound constraints. In this paper, a one-layer recurrent neural network is proposed for solving non-smooth convex programming problem (1). Comparing with the existing neural networks for non-smooth convex optimization, this new neural network has a simpler structure, but can be utilized to solve more general convex programming problems.

2 Model Description

In this section, a one-layer recurrent neural network is constructed for solving programming problem (1).

Definition 1. [17] *Suppose $E \subset \mathbb{R}^n$. $F : x \mapsto F(x)$ is called a set-valued function from $E \hookrightarrow \mathbb{R}^n$, if to each point x of a set E, there corresponds to a nonempty closed set $F(x) \subset \mathbb{R}^n$.*

Definition 2. [18] *Let $V(x)$ be a function from $\mathbb{R}^n$ to $\mathbb{R}$. For any $x \in \mathbb{R}^n$,*

$$DV(x)(v) = \lim_{h \to 0+} \frac{V(x+hv) - V(x)}{h}.$$

We say that $DV(x)(v)$ is the derivative from the right of V at x in the direction v. If $DV(x)(v)$ exists for all directions, we say that V is differentiable from the right at x. We say that the closed convex subset (possibly empty)

$$\partial V(x) = \{\xi \in \mathbb{R}^n : \forall v \in \mathbb{R}^n, \xi^T v \le DV(x)(v)\}$$

is the sub-differential of V at x. The element ξ of $\partial V(x)$ is called the sub-gradient of V at x.

Theorem 1. *x^* is an optimal solution of problem (1) if and only if there exists $y^* \in \mathbb{R}^m$ such that $({x^*}^T, {y^*}^T)^T$ satisfies the following equations*

$$0 \in \partial f(x) - A^T y, \tag{2}$$

$$0 = Ax - b. \tag{3}$$

Proof: See Theorem 1 in [16]. □

Next, according to Equations (2) and (3), the one-layer recurrent neural network model will be induced.

From (2), for any $\gamma \in \partial f(x)$, we have

$$x = x - \gamma + A^T y. \tag{4}$$

Substituting (4) into (3), it follows that

$$A(x - \gamma + A^T y) - b = 0, \quad \forall \gamma \in \partial f(x).$$

That is

$$AA^T y = A\gamma - Ax + b, \quad \forall \gamma \in \partial f(x).$$

Since A is full raw-rank, AA^T is invertible. Then

$$y = (AA^T)^{-1}(A\gamma - Ax + b), \quad \forall \gamma \in \partial f(x). \tag{5}$$

Substituting (5) into (2), for any $\gamma \in \partial f(x)$, we have

$$\gamma - A^T(AA^T)^{-1}(A\gamma - Ax + b) = 0. \tag{6}$$

Let $P = A^T(AA^T)^{-1}A$ and $q = A^T(AA^T)^{-1}b$, then, (6) can be written as

$$Px + (I - P)\gamma - q = 0, \quad \forall \gamma \in \partial f(x), \tag{7}$$

where I is identity matrix. The matrix P, called the projection matrix, is symmetric and satisfies $P^2 = P$.

Based on Equation (7), the proposed recurrent neural network model is described by the following differential inclusion:

$$\frac{dx}{dt} \in \lambda[-Px - (I - P)\partial f(x) + q], \tag{8}$$

where λ is a positive scaling constant.

Definition 3. *x^* is said to be an equilibrium point of neural network (8) if there exists $\gamma^* \in \partial f(x^*)$ such that*

$$-Px^* - (I - P)\gamma^* + q = 0. \tag{9}$$

From above analysis, the following theorem obviously holds.

Theorem 2. *x^* is an optimal solution of problem (1) if and only if it is an equilibrium point of neural network (8).*

3 Global Convergence

In this section, the global convergence of the recurrent neural network (8) is analyzed. Throughout this paper, we always assume that the optimal solution set (denoted as Ω^*) of problem (1) is not empty and there always exists a finite $x^* \in \Omega^*$. Then the equilibrium point set (denoted as Ω^e) of neural network (8) is nonempty.

Definition 4. *The neural network* (8) *is said to be globally convergent to an optimal solution of problem* (1) *if for any trajectory $x(t)$ of the neural network with initial point $x(t_0) \in \mathbb{R}^n$, there exists an optimal solution $x^* \in \Omega^*$ such that $\lim_{t\to+\infty} x(t) = x^*$.*

Theorem 3. *The neural network* (8) *is stable in the sense of Lyapunov and globally convergent to an optimal solution of problem* (1).

Proof: Assume x^* being an equilibrium point of neural network (8), then there exists $\gamma^* \in \partial f(x^*)$ such that

$$-Px^* - (I-P)\gamma^* + q = 0.$$

By substituting it into Equation (8), it can be rewritten as

$$\frac{dx}{dt} \in \lambda[-P(x-x^*) - (I-P)(\partial f(x) - \gamma^*)]. \tag{10}$$

Consider the following Lyapunov function

$$V(x) = \lambda[f(x) - f(x^*) - (x-x^*)^T\gamma^* + \frac{1}{2}\|x-x^*\|_2^2], \tag{11}$$

We have

$$\partial V(x) = \lambda[\partial f(x) - \gamma^* + x - x^*].$$

By using the chain rule [19], it follows that $V(x(t))$ is differentiable for a.a. $t \geq 0$ and it results in

$$\dot{V}(z(t)) = \xi(t)^T\dot{x}(t), \quad \forall \xi(t) \in \partial V(x(t)).$$

Let

$$\xi(t) = \lambda[\gamma(t) - \gamma^* + x(t) - x^*],$$

where $\gamma(t) \in \partial f(x(t))$. Then

$$\begin{aligned}\dot{V}(x(t)) &\leq \lambda^2 \sup_{\gamma\in\partial f(x)} [\gamma-\gamma^*+x-x^*]^T[-P(x-x^*)-(I-P)(\gamma-\gamma^*)]\\ &= \lambda^2 \sup_{\gamma\in\partial f(x)} [-(\gamma-\gamma^*)^T(I-P)(\gamma-\gamma^*) - (x-x^*)^TP(x-x^*)\\ &\quad -(x-x^*)^T(\gamma-\gamma^*)]. \end{aligned} \tag{12}$$

Since $f(x)$ is convex, for any $\gamma \in \partial f(x)$, $(x-x^*)^T(\gamma-\gamma^*) \geq 0$ holds. Then

$$\begin{aligned}\dot{V}(x(t)) &\leq \lambda^2 \sup_{\gamma\in\partial f(x)} [-(\gamma-\gamma^*)^T(I-P)(\gamma-\gamma^*) - (x-x^*)^T P(x-x^*)] \\ &= -\lambda^2 \inf_{\gamma\in\partial f(x)} [(\gamma-\gamma^*)^T(I-P)(\gamma-\gamma^*) + (x-x^*)^T P(x-x^*)]. \quad (13)\end{aligned}$$

On the other hand, for any $\gamma \in \partial f(x)$,

$$\begin{aligned}\|\dot{x}\|_2^2 &= \lambda^2[-P(x-x^*) - (I-P)(\gamma-\gamma^*)]^T[-P(x-x^*) - (I-P)(\gamma-\gamma^*)] \\ &= \lambda^2[(x-x^*)^T P(x-x^*) + (\gamma-\gamma^*)^T(I-P)(\gamma-\gamma^*)]. \quad (14)\end{aligned}$$

From (13) and (14), it follows that

$$\begin{aligned}\dot{V}(x(t)) &\leq -\inf_{\gamma\in\partial f(x)} \|\dot{x}\|_2^2 \\ &= -\lambda^2 \inf_{\gamma\in\partial f(x)} \|Px + (I-P)\gamma - q\|_2^2. \quad (15)\end{aligned}$$

From (11), we have $V(x) \geq \lambda\|x-x^*\|_2^2/2$. Let $L(x_0) = \{x \in \mathbb{R}^n : V(x) \leq V(x_0)\}$, then $L(x_0)$ is bounded. From (15), $x(t)$ is also bounded and it follows that the solution $x(t)$ exists on $[t_0, +\infty)$. $V(x)$ is a Lyapunov function of (8) and neural network (8) is stable in the sense of Lyapunov.

Define $\Gamma(x) = \inf_{\gamma\in\partial f(x)} \|Px+(I-P)\gamma-q\|_2^2$. If $x^* \in \Omega^e$, we have $\Gamma(x^*) = 0$. Conversely, if there exists $\hat{x} \in \mathbb{R}^n$ such that $\Gamma(\hat{x}) = 0$, combining that $\partial f(x)$ is a compact convex subset in $\mathbb{R}^n$, then there exists $\hat{\gamma} \in \Gamma(\hat{x})$ such that

$$P\hat{x} + (I-P)\hat{\gamma} - q = 0.$$

Therefore, $\Gamma(x) = 0$ if and only if $x \in \Omega^e$.

As the rest proof is similar to that of Theorem 1 in [10], it is omitted here. □

4 Simulation Results

In this section, two examples are given to demonstrate the effectiveness of the recurrent neural network proposed in this paper for solving the constrained least absolute deviation and nonlinear curve-fitting problems.

Example 1. Consider the following constrained least absolute deviation problem:

$$\begin{array}{ll}\text{minimize} & \|Cx-d\|_1, \\ \text{subject to} & Ax = b,\end{array} \quad (16)$$

where $x = (x_1, x_2, x_3, x_4)^T$ and

$$C = \begin{pmatrix} 4 & 2 & -1 & 2 \\ 1 & 3 & 2 & -1 \end{pmatrix}, d = \begin{pmatrix} -3 \\ 5 \end{pmatrix}, A = \begin{pmatrix} 4 & 1 & -2 & 1 \\ 1 & 3 & 1 & -1 \end{pmatrix}, b = \begin{pmatrix} -2 \\ 4 \end{pmatrix}.$$

This problem has a unique optimal solution $x^* = (0.5, 0.125, 1, -2.125)^T$. Let $\lambda = 10^6$, the simulation results are shown in Fig. 1 with 10 random initial values. We can see that the state trajectories of neural network (8) is globally convergent to the unique optimal solution.

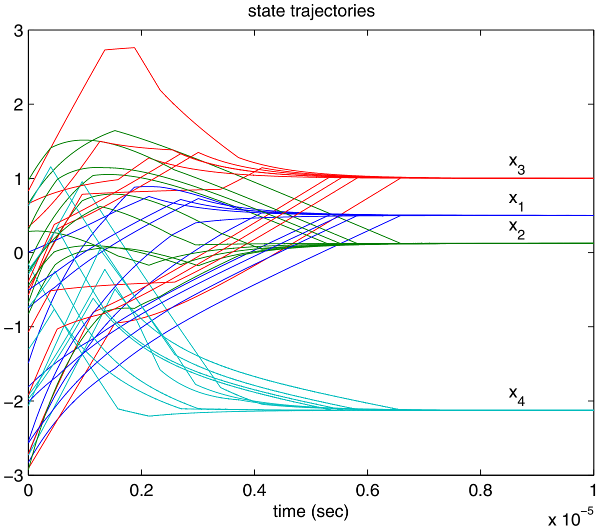


Fig. 1. Transient behavior of the neural network (8) in Example 1

Example 2. Nonlinear Curve-Fitting Problem
Let us consider a constrained nonlinear least absolute deviation curve-fitting problem: Find the parameters of the combination of exponential and polynomial function $y(x) = a_4 e^x + a_3 x^3 + a_2 x^2 + a_1 x + a_0$, which fits the data given in Table 1 and subjects to the equalities $y(1.2) = -7.8$ and $y(4.6) = -3.4$. This problem can be formulated as follows:

$$\begin{array}{ll} \text{minimize} & \|Cx - d\|_1, \\ \text{subject to} & Ax = b, \end{array} \tag{17}$$

where $x = (x_1, x_2, x_3, x_4, x_5)^T = (a_4, a_3, a_2, a_1, a_0)^T$ and

$$C = \begin{pmatrix} 1 & 1.649 & 2.718 & 4.482 & 7.389 & 12.183 & 20.086 & 33.116 & 54.598 & 90.017 \\ 0 & 0.125 & 1 & 3.375 & 8 & 15.625 & 27 & 42.875 & 64 & 91.125 \\ 0 & 0.25 & 1 & 2.25 & 4 & 6.25 & 9 & 12.25 & 16 & 20.25 \\ 0 & 0.5 & 1 & 1.5 & 2 & 2.5 & 3 & 3.5 & 4 & 4.5 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}^T,$$

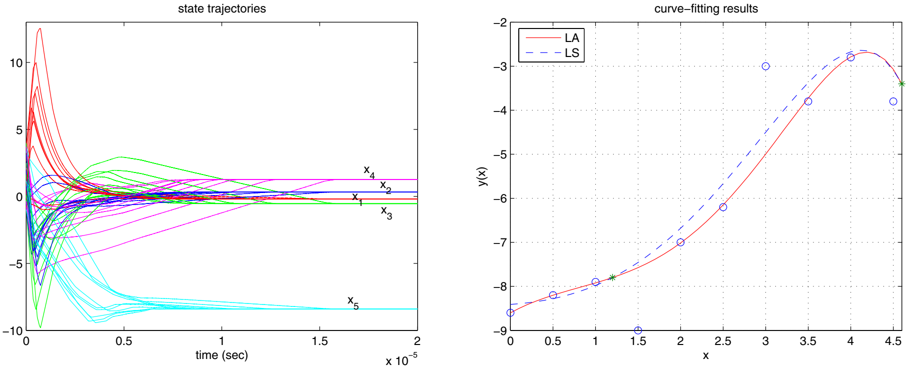
$$d = \begin{pmatrix} -8.6 & -8.2 & -7.9 & -9 & -7 & -6.2 & -3 & -3.8 & -2.8 & -3.8 \end{pmatrix}^T,$$

$$A = \begin{pmatrix} 3.32 & 1.728 & 1.44 & 1.2 & 1 \\ 99.484 & 97.336 & 21.16 & 4.6 & 1 \end{pmatrix}, b = \begin{pmatrix} -7.8 \\ -3.4 \end{pmatrix},$$

By utilizing the neural network in (8) for solving this problem, the simulation results are shown in Fig. 2(a) with $\lambda = 10^6$ and 10 random initial points, from which we can see that the neural network is globally convergent to the optimal

Table 1. Nonlinear fitting data for Example 2

x	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5
y	-8.6	-8.2	-7.9	-9	-7	-6.2	-3	-3.8	-2.8	-3.8



(a)

(b)

Fig. 2. (a) Transient behavior of neural network (8) in Example 2; (b) Comparison of the two nonlinear curve fitting methods between the LA and LS in Example 2

solution $x^* = (-0.21, 0.32, -0.55, 1.26, -8.39)^T$ or $(a_4, a_3, a_2, a_1, a_0) = (-0.21, 0.32, -0.55, 1.26, -8.39)$. The curve fitting is drawn in Fig. 2(b) for l_1-norm (solid line) and l_2-norm (dashed line). It shows that least absolute (LA) has better fitting performance than least square (LS) in dealing with outliers.

5 Conclusions

In this paper, a one-layer recurrent neural network has been proposed for solving non-smooth convex programming problems with linear equality constraints. Comparing with other neural networks for convex optimization, the proposed neural network has lower architecture complexity with only one-layer structure and less neurons. However, it is efficient for solving both smooth and non-smooth optimization problems. The global convergence of the neural network are proved based on the Lyapunov theory. Furthermore, the proposed recurrent neural network is efficient for solving the constrained least absolute deviation and nonlinear curve-fitting problems. Simulation results show the performance and effectiveness of the proposed neural network.

References

1. Tank, D., Hopfield, J.: Simple neural optimization networks: An a/d converter, signal decision circuit, and a linear programming circuit. IEEE Transactions on Circuits and Systems 33, 533–541 (1986)

2. Kennedy, M., Chua, L.: Neural networks for nonlinear programming. IEEE Transactions on Circuits and Systems 35, 554–562 (1988)
3. Wang, J.: Analysis and design of a recurrent neural network for linear programming. IEEE Transactions on Circuits and Systems-I 40, 613–618 (1993)
4. Wang, J.: A deterministic annealing neural network for convex programming. Neural Networks 7, 629–641 (1994)
5. Wang, J., Xia, Y.: Analysis and design of primal-dual assignment networks. IEEE Transactions on Neural Networks 9, 183–194 (1998)
6. Xia, Y., Wang, J.: Neural network for solving least absolute and related problems. Neurocomputing 19, 13–21 (1998)
7. Xia, Y., Wang, J.: A general projection neural network for solving monotone variational inequalities and related optimization problems. IEEE Transactions on Neural Networks 15, 318–328 (2004)
8. Yang, Y., Cao, J.: Solving quadratic programming problems by delayed projection neural network. IEEE Transactions on Neural Networks 17, 1630–1634 (2006)
9. Hu, X., Wang, J.: Solving pseudomonotone variational inequalities and pseudoconvex optimization problems using the projection neural network. IEEE Transactions on Neural Networks 17, 1487–1499 (2006)
10. Liu, Q., Wang, J.: A one-layer recurrent neural network with a discontinuous hard-limiting activation function for quadratic programming. IEEE Transactions on Neural Networks 19, 558–570 (2008)
11. Zhang, S., Constantinides, A.: Lagrange programming neural networks. IEEE Transactions on Circuits and Systems-II 39, 441–452 (1992)
12. Xia, Y., Wang, J.: On the stability of globally projected dynamical systems. Journal of Optimization Theory and Applications 106, 129–150 (2000)
13. Xia, Y., Leung, H., Wang, J.: A projection neural network and its application to constrained optimization problems. IEEE Transactions Circuits and Systems-I 49, 447–458 (2002)
14. Liu, Q., Wang, J.: A one-layer recurrent neural network with a discontinuous activation function for linear programming. Neural Computation 20, 1366–1383 (2008)
15. Li, G., Song, S., Wu, C., Du, Z.: A neural network model for non-smooth optimization over a compact convex subset. In: Wang, J., Yi, Z., Żurada, J.M., Lu, B.-L., Yin, H. (eds.) ISNN 2006. LNCS, vol. 3971, pp. 344–349. Springer, Heidelberg (2006)
16. Liu, Q., Wang, J.: A recurrent neural network for non-smooth convex programming subject to linear equality and bound constraints. In: King, I., Wang, J., Chan, L.-W., Wang, D. (eds.) ICONIP 2006. LNCS, vol. 4233, pp. 1004–1013. Springer, Heidelberg (2006)
17. Filippov, A.: Differential Equations with Discontinuous Righthand Sides. Mathematics and its applications (Soviet series). Kluwer Academic Publishers, Boston (1988)
18. Aubin, J., Cellina, A.: Differential Inclusions: Set-Valued Maps and Viability Theory. Springer, New York (1984)
19. Forti, M., Nistri, P.: Global convergence of neural networks with discontinuous neuron activations. IEEE Transactions on Circuits and Systems-I 50, 1421–1435 (2003)

Part VIII

Brain-Computer Interface

A Study on Application of Reliability Based Automatic Repeat Request to Brain Computer Interfaces

Hiromu Takahashi, Tomohiro Yoshikawa, and Takeshi Furuhashi

Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan
takahashi@cmplx.cse.nagoya-u.ac.jp,
{yoshikawa,furuhashi}@cse.nagoya-u.ac.jp

Abstract. Recently, a lot of research on a Brain Computer Interface (BCI) which enables patients like those with Amyotrophic Lateral Sclerosis to control some equipment or to communicate with other people has been reported. One of the problems in BCI research is a trade-off between the speed and the accuracy. In the field of data transmission, on the other hand, Reliability-Based Hybrid ARQ (RB-HARQ) has been developed to achieve both of the performances. In this paper, therefore, BCIs are considered as communications between users and computers, and Reliability-Based ARQ, similar to RB-HARQ, is applied to BCIs. Through simulations and experiments, it is shown that the proposed method is superior to other methods.

1 Introduction

Recently, a lot of research on a Brain Computer Interface (BCI) which records brain activities, discriminates the thoughts, and then enables patients like those with Amyotrophic Lateral Sclerosis (ALS) to control some equipment or to communicate with others has been reported. The authors also have been studying on a BCI based on Electroencephalogram (EEG), which is considered as one of the most reasonable measurements since it is non-invasive and costs less [1]. In fact, EEG-based BCIs have been researched well; for example, Thought Translations Device (TTD) [2] by Birbaumer et al.; and Graz-BCI [3] by Pfurtscheller et al. [3], which employs the band power from 8 to 13 Hz (alpha band) as the feature of EEG, and applies Linear Discriminant Analysis (LDA) [4] to it. The accuracy, however, is not high since biological signals such as EEGs contain much noise, partly due to users' physical or mental conditions. On the other hand, it also has been suggested that the longer EEG data is used for one discrimination, the more accuracy could be achieved [3,5,6]. It could be possible to say that high accuracy can be gained in exchange for speed in those methods; here seems a trade-off between the accuracy and the speed. The purpose of this study, therefore, is to develop a BCI which accomplishes both the accuracy and the speed simultaneously.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 1013–1020, 2009.

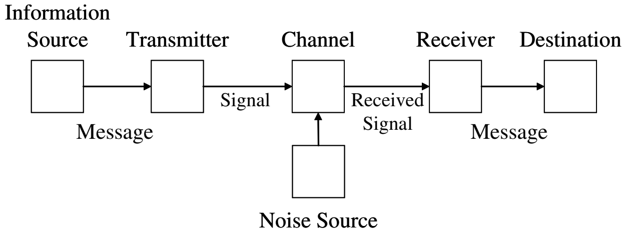


Fig. 1. Shannon's Communication Model

In the field of data transmission, there are some error control methods; for instance, Forward Error Correction (FEC), which allows the receiver to detect and correct errors; Automatic Repeat reQuest (ARQ), which asks the transmitter to repeat code words; and Hybrid ARQ (HARQ), which is a combination of ARQ and FEC. In the past several years, Reliability-Based Hybrid ARQ (RB-HARQ) has been proposed [7]. This method is a variation of HARQ, in which the requests are based on reliability of each bit in code words. It also has been reported that RB-HARQ can provide performance close to the channel capacity. In this paper, BCIs are considered as communications between users and computers or other people, are modeled using Shannon's communication model, and then Reliability-Based ARQ (RB-ARQ) is applied to BCIs. This paper compares the proposed method with other possible methods and it shows that the proposed method is effective for BCIs in terms of both the accuracy and the speed.

2 Proposed Method

2.1 Modeling of BCIs

In EEG-based BCIs, firstly scalp potentials, which reflect the users' will, are recorded. Then the recorded EEG data is classified by statistical classifiers such as LDA, and translated into commands to control some equipment or to communicate with others. In these processes, distractions such as hunger, sleepiness, fatigue, and electric noises would affect the EEG data, which leads misclassifications or mistranslations as a result. Figure 1 shows Shannon's Communication Model. BCIs could be regarded as communications between users as information source and computers as destination when nerves, electrodes, cables, and classifiers are regarded as channels. Note that users also take a role of a transmitter and computers take that of a receiver in Fig. 1.

2.2 Applying RB-ARQ to BCI

Automatic Repeat reQuest (ARQ) is an error control method for data transmission, in which the receiver requests the transmitter to send the data again if errors are detected. In this paper, a method requesting a retransmission based on the reliability is called RB-ARQ, and the RB-ARQ is applied to BCIs. Note that

RB-ARQ does not include FEC because the process of FEC would be difficult for users, who take the role of the transmitter.

Suppose users have one thought in their mind out of two (i.e. binary selection), and p-dimensional feature vector is obtained from EEG data. Corresponding to these, let $u_i \in \{0, 1\}$ be the ith information bit and $\boldsymbol{y}_i^t \in \mathbb{R}^p$ be the received analog information at time t. Suppose $f_k(\boldsymbol{y})$ is the class-conditional probability density function of $\boldsymbol{y}$ in $u_i = k$. Then, the log-likelihood ratio can be obtained as follows:

$$\lambda_i^t = \ln \frac{Pr\left(u_i = 0 | \boldsymbol{y}_i^1, \ldots, \boldsymbol{y}_i^t\right)}{Pr\left(u_i = 1 | \boldsymbol{y}_i^1, \ldots, \boldsymbol{y}_i^t\right)} \tag{1a}$$

$$= \sum_{j=1}^{t} \ln \frac{f_0(\boldsymbol{y}_i^j)}{f_1(\boldsymbol{y}_i^j)}. \tag{1b}$$

$|\lambda_i^t|$ represents the reliability; the larger it is, the higher the probability of correct decoding is. Hence, when a certain λ is given and $|\lambda_i^t| < \lambda$ is true, the receiver requests the same information and decodes it again; otherwise the ith information bit can be estimated as follows:

$$\hat{u}_i = \begin{cases} 0 & \lambda_i^t \geq 0 \\ 1 & \textit{otherwise}. \end{cases} \tag{2}$$

The proposed method assumes that $f_0(\boldsymbol{y}), f_1(\boldsymbol{y})$ are p-dimensional Gaussian distributions with mean vector: $\boldsymbol{\mu}_0, \boldsymbol{\mu}_1$, covariance matrix: $\Sigma_0 = \Sigma_1 = \Sigma$, because of their simplicity and less computational cost. Especially when $\lambda = 0$, this is identical to LDA. Equation (3) is called discriminability, representing how easily the data can be discriminated [8],

$$d = \frac{|\boldsymbol{\mu}_0^T \boldsymbol{w} - \boldsymbol{\mu}_1^T \boldsymbol{w}|}{\sqrt{\boldsymbol{w}^T \Sigma \boldsymbol{w}}}, \tag{3}$$

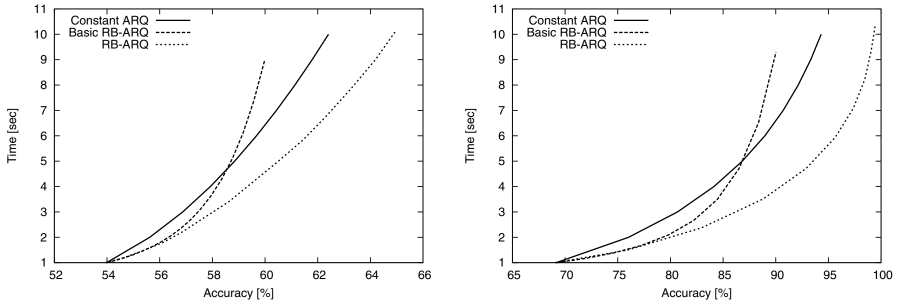
where $\boldsymbol{w} = \Sigma^{-1}(\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1)$.

2.3 Comparison of Methods in Theoretical Value

The proposed method was compared with two possible methods. The first one can be called Constant ARQ requesting n times constantly regardless of the reliability. The second one can be called Basic RB-ARQ taking $\lambda^{(t)}$ obtained from (4).

$$\lambda_i^t = \ln \frac{Pr\left(u_i = 0 | \boldsymbol{y}_i^t\right)}{Pr\left(u_i = 1 | \boldsymbol{y}_i^t\right)} \tag{4}$$

Suppose both $f_0(\boldsymbol{y})$ and $f_1(\boldsymbol{y})$ obey p-dimensional Gaussian distributions with $\Sigma = \Sigma_0 = \Sigma_1$. Figure 2 describes the relationships between the accuracy and the transmission time (i.e. speed), comparing the proposed method (RB-ARQ) with the two methods mentioned above. The figure shows that the longer the transmission time is, the better the accuracy is. It also clearly shows that RB-ARQ is superior to the others in terms of both performances.



(a) $d = 0.2$ (b) $d = 1$

Fig. 2. Comparison of Methods (Theoretical Value)

3 Experiments

3.1 Experimental Settings

According to the international 10-20 system [9], electrodes were places on Pz, and A2 as a reference. Then, EEGs were recorded with 1000 Hz sampling rate using Polymate AP216 manufactured by Degitex co, ltd. One trial consisted of 10-second measurement and 10-second break, and one run consisted of 12 trials. Four runs in one session, which were two runs for being relax called thought A and two runs for mental arithmetic called thought B, were performed by each subject. Six subjects in their early 20's participated in two sessions, however one session was excluded out of 12 sessions because of a lack of data. One dataset was 1-second EEG data; therefore 480 datasets were obtained for each session. The Fast Fourier Transform (FFT) was applied to the datasets and they were transformed into the band-power of alpha band (i.e. from 8 to 13 Hz).

3.2 Test of Fitness to Gaussian Distribution

In the proposed method, a band power of EEG is assumed to be normally distributed. In order to verify this assumption, the chi-square goodness of fit test was applied to the band power from 8 to 13 Hz, and the null hypothesis that it was normally distributed was not rejected at statistically significant level of 0.05. Figure 3 shows the histograms of a band power from 8 to 13 Hz when a subject had thought A and B in his mind, and the Gaussian distributions corresponding to the data, respectively.

3.3 Application of RB-ARQ

The averages and variances were estimated using the first half 240 datasets (120 datasets from each thought) as training data (note that equal variances were assumed). Then, the rest 240 datasets were applied to the three methods as test data as follows:

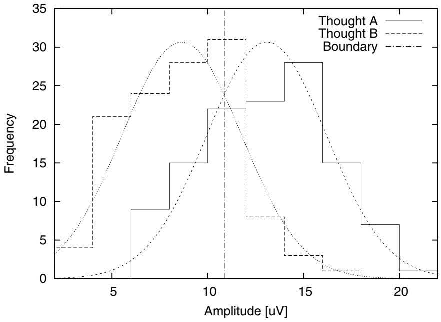


Fig. 3. Histogram of Band Power from 8 to 13 Hz

Step 1. Set $l = i = t = 1$.

Step 2. Let x_l be the lth dataset and $\boldsymbol{y}_i^t$ be the received analog information of the ith information bit at time t corresponding to x_l

Step 3. Calculate λ_i^t based on (1a)

Step 4. If $|\lambda_i^t| < \lambda$, add 1 to both l and t, and go to step 2; otherwise go to the next step

Step 5. Estimate $\hat{u}_i$, add 1 to both i and l, substitute T_i (the transmission time for ith information bit) for t, t for one, and go to step 2

After these procedures, let N_V be the number of information bits, N_C be the number of them which fulfill $\hat{u}_i = u_i$. The accuracy and the transmission time are defined as follows:

$$Accuracy[\%] = N_\mathrm{C}/N_\mathrm{V} \times 100, \tag{5}$$

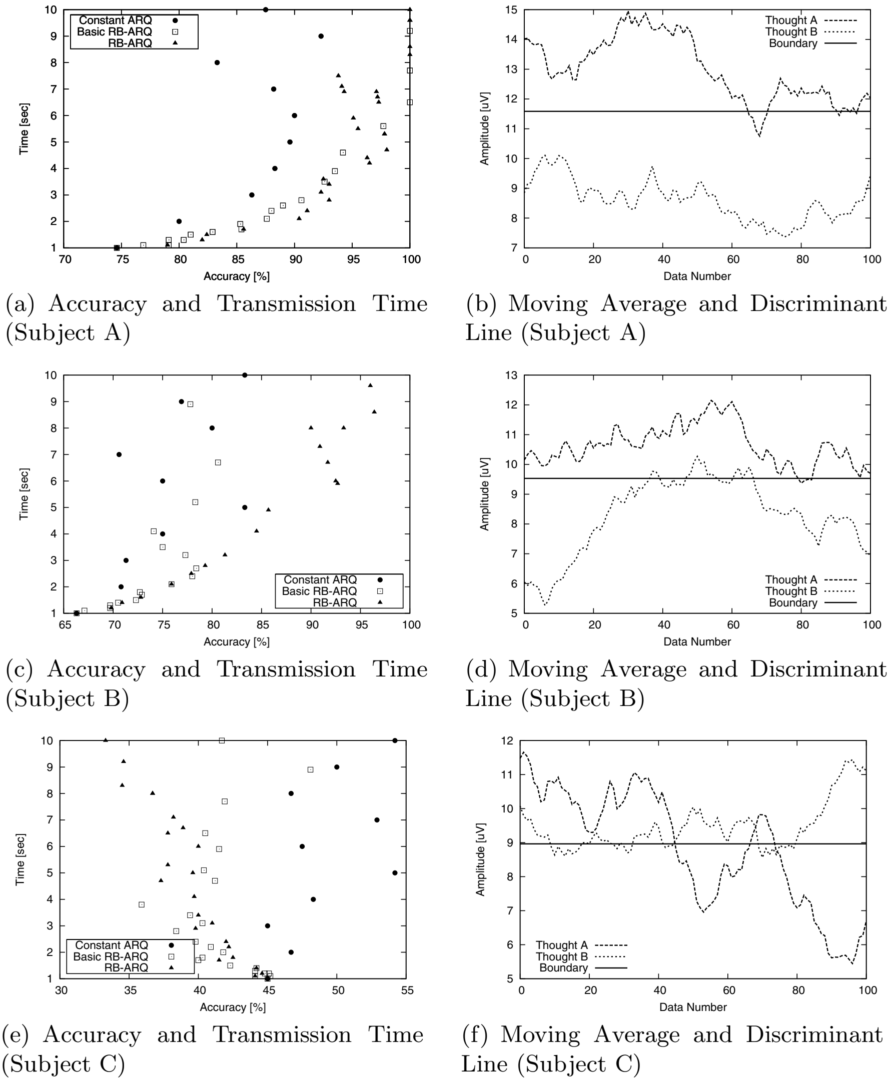
$$Time[sec] = \sum_{i=1}^{N_\mathrm{V}} T_i/N_\mathrm{V}. \tag{6}$$

Figure 4 shows the relationships between the accuracies and the transmission times at certain λ (for RB-ARQ and Basic RB-ARQ) or n (for Constant ARQ) along with figures showing the 20 datasets' simple moving averages and the discrimination boundaries (i.e. λ_i^t=0).

4 Discussions

Firstly, the test of fitness could not reject the null hypothesis that the data used in this experiment did not follow the Gaussian distribution; therefore, it is not unreasonable to employ the Gaussian distribution. However, it does not mean that the data follow it. To make the distribution more closed to the Gaussian, it might be better to use the logarithm of the band power [10].

Figure 4(a) shows a case where the performance of RB-ARQ was roughly equal to that of Basic RB-ARQ and both of them were superior to Constant ARQ. In this case, the assumption about distributions was reasonable because



(a) Accuracy and Transmission Time (Subject A)

(b) Moving Average and Discriminant Line (Subject A)

(c) Accuracy and Transmission Time (Subject B)

(d) Moving Average and Discriminant Line (Subject B)

(e) Accuracy and Transmission Time (Subject C)

(f) Moving Average and Discriminant Line (Subject C)

Fig. 4. Comparison of Methods (Experimental Value)

of the result of the fitness test and that the averages and the variance of learning data, $\mu_0 = 3.51, \mu_1 = 3.18, \sigma = 3.35$, were nearly equal to those of test data, $\mu_0 = 3.34, \mu_1 = 2.88, \sigma = 3.35$. RB-ARQ, therefore, should be superior to the other two methods theoretically as shown in Fig. 2.

Figure 4(c) shows another case where RB-ARQ was superior to the rest. According to Fig. 4(d), it can be seen that the moving average of thought B moved dynamically. The datasets from data sequence 0 to 20 could be classified correctly without combining datasets, in other words, requesting next dataset, because distances between the moving average and the boundary was large;

requesting a certain number of datasets, Constant ARQ has a disadvantage in terms of transmission time in those datasets. Around data sequence 30 to 60, it was expected to discriminate them correctly by combining datasets because the moving average of thought B was close to the boundary; not combining datasets, Basic RB-ARQ has a disadvantage in terms of accuracy. On the other hand, RB-ARQ has an advantage especially when the moving average moves like this case because it classifies combined datasets based on the reliability, i.e. the distance from the boundary. Some methods which adjust the discrimination models since the optimal model would change as time passes because of the instability of EEGs [11,12] have been proposed so far. These reports indicate that the proposed method could be improved by adjusting the averages or the variances in response to the changing EEGs.

Figure 4(e) also shows another case where Constant ARQ was superior to the others. Since the magnitude relation between averages of thought A and B in test data was different from that in learning data, accuracies were below 50 %. According to Fig. 4(f), the reverse of the magnitude relation is obvious especially after data sequence 80.

5 Summary

In this paper, BCIs were regarded as communications between users and computers based on Shannon's communication model, and a thought recognition method based on RB-ARQ was proposed in order to improve the speed and the accuracy simultaneously. Some simulations and experiments were performed, which showed that the proposed method was superior to other possible methods. In some of experiments, however, the average of the EEG's band power had changed through the experiment, which lowered the accuracies. A method which is adaptive to the changes, therefore, seems necessary to make the performance better.

References

1. Tateoka, Y., Yoshikawa, T., Furuhashi, T., Tanaka, K.: A basic study on electroencephalogram-based control. In: SCIS&ISIS 2006, pp. 1959–1962 (2006)
2. Birbaumer, N., Kubler, A., Ghanayim, N., Hinterberger, T., Perelmouter, J., Kaiser, J., Iversen, I., Kotchoubey, B., Neumann, N., Flor, H.: The thought translation device (ttd) for completely paralyzed patients. IEEE Trans. on Rehabilitation Engineering 8(2), 190–193 (2000)
3. Scherer, R., Muller, G., Neuper, C., Graimann, B., Pfurtscheller, G.: An asynchronously controlled eeg-based virtual keyboard: improvement of the spelling rate. IEEE Trans. on Biomedical Engineering 51(6), 979–984 (2004)
4. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer, Heidelberg (2001)
5. Santhanam, G., Ryu, S.I., Yu, B.M., Afshar, A., Shenoy, K.V.: A high-performance brain-computer interface. Nature 442, 195–198 (2006)

6. McFarland, D.J., Sarnacki, W.A., Wolpaw, J.R.: Brain-computer interface (bci) operation: optimizing information transfer rates. Biological Psychology 63, 237–251 (2003)
7. Shea, J.M.: Reliability-based hybrid arq. IEE Electronics Letters 38(13), 644–645 (2002)
8. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. In: New Technology Communications, 2nd edn. (2001)
9. Jasper, H.H.: The ten twenty electrode system of the international federation. Electroencephalography and Clinical Neurophysiology 10, 371–375 (1958)
10. Gasser, T., Bacher, P., Mocks, J.: Transformations towards the normal distribution of broad band spectral parameters of the eeg. Electroencephalography and clinical neurophysiology 53, 119–124 (1982)
11. Vidaurre, C., Schlogl, A., Cabeza, R., Scherer, R., Pfurtscheller, G.: A fully on-line adaptive bci. IEEE Trans. on Biomedical Engineering 53(6), 1214–1219 (2006)
12. Sykacek, P., Roberts, S., Stokes, M.: Adaptive bci based on variational bayesian kalman filtering: an empirical evaluation. IEEE Trans. on Biomedical Engineering 51(5), 719–727 (2004)

Analysis on Saccade-Related Independent Components by Various ICA Algorithms for Developing BCI

Arao Funase[1,2], Motoaki Mouri[1,2], Yagi Tohru[3,4], Andrzej Cichocki[2], and Ichi Takumi[1]

[1] Graduate School of Engineering, Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya, 466-8555, Japan
[2] Brain Science Institute, RIKEN, 2-1, Hirosawa, Wako, 351-0198, Japan
[3] Graduate School of Information Science and Engineering, Tokyo Institute of Technology, 1-12-1, Ookayama, Meguro-ku, Tokyo, 152-8555, Japan
[4] Bio-Mimetic Control Research Center, RIKEN 2271-130, Anagahora, Shimoshidami, Moriyama-ku, Nagoya, 463-0003, Japan

Abstract. Saccade-related electroencephalogram (EEG) signals have been the subject of application oriented research by our group toward developing a brain computer interface (BCI). Our goal is to develop novel BCI based on eye movements system employing EEG signals on-line. Most of the analysis of the saccade-related EEG data has been performed using ensemble averaging approaches. In signal processing method for BCI, raw EEG signals are analyzed. In ensemble averaging method which is major EEG analysis is not suitable for processing raw EEG signals. In order to process raw EEG data, we use independent component analysis. This paper presents extraction rate of saccade-related EEG signals by four ICA algorithms and six window size. In terms of extracting rates across ICA algorithms, The JADE and Fast ICA have good results. As you know, calculation time in Fast ICA is faster than calculation time in JADE. Therefore, in this case, Fast ICA is the best in order to extract saccade-related ICs. Next, we focus on extracting rates in each window. The windows not including EEG signals after saccade and the windows which has small window size has better extracting rates.

1 Introduction

Brain-computer interfaces (BCIs) have been the subject of research efforts for several decades [1][2]. The capabilities of BCIs allow them to be used in situations unsuitable for the conventional interfaces. BCIs are used to connect a user and a computer via an electroencephalogram (EEG). The EEG is related to emotion, motion, and thought. Therefore, there is the potential that BCIs can be used to connect normal and mobility-impaired persons to computers in such a way that no movement on the part of the user is required. Moreover, the *Quality of Life*

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 1021–1028, 2009.

for severely handicapped users is expected to be improved by using BCIs to connect these users to computers.

EEG related to saccadic eye movement have been studied by our group toward developing a BCI eye-tracking system operated by using saccade-related EEG [3]. In previous researches, EEG data were analyzed using the ensemble averaging method. Ensemble averaging is not suitable for analyzing raw EEG data because the method needs many repetitive trials.

Recording EEG data repetitively is a critical problem to develop BCIs. Overcoming this critical problem is essential to realize a practical use of BCIs for single trial EEG data.

Recently, the independent component analysis (ICA) method has been introduced in the field of bio-signal processing as a promising technique for separating independent sources. The ICA method can process raw EEG data and find features. Therefore, ICA overcomes the problems associated with ensemble averaging, and it observes the waveforms of the EEG data.

There are many algorithms that are used in the field of ICA[4]. In previous researches, I analyzed saccade-related independent components by only one algorithms; FastICA. However, it is important to check which ICA algorithms have best performance for extracting saccade-related EEG signals.

In order to extract independent components from EEG signals, we must decide a window size to analyze EEG signals. It is important to decide an appropriate window size to get good independent components.

In this paper, we pick up four algorithms; FastICA[5], NG-ICA[7], AMUSE[6], JADE[8]. We check which algorithms can get the best extracting rate of saccade-related independent components and what window size have good performance to extract saccade-related independent components.

2 Independent Component Analysis (ICA)

The ICA method is based on the following principle (See Fig. 1). Assuming that the original (or source) signals have been linearly mixed, and that these mixed signals are available, ICA recognises in a blind manner a linear combination of the mixed signals, and recovers the original source signals, possibly re-scaled and randomly arranged in the outputs.

The $\mathbf{s} = [s_1, s_2, \cdots, s_n]^{\mathbf{T}}$ means n independent signals from mutual EEG sources in the brain, for example. The mixed signals $\mathbf{x}$ are thus given by $\mathbf{x} = \mathbf{A}\mathbf{s}$, where $\mathbf{A}$ is an $n \times n$ invertible matrix. $\mathbf{A}$ is the matrix for mixing independent signals. In the ICA method, only $\mathbf{x}$ is observed. The value for $\mathbf{s}$ is calculated by $\mathbf{s} = \mathbf{W}\mathbf{x}$ ($\mathbf{W} = \mathbf{A}^{-1}$). However, it is impossible to calculate $\mathbf{A}^{-1}$ algebraically because information for $\mathbf{A}$ and $\mathbf{s}$ are not already known. Therefore, in the ICA algorithm, $\mathbf{W}$ is estimated non-algebraically. The assumption of the ICA algorithm is that $\mathbf{s}$ is mutually independent. In order to calculate $\mathbf{W}$, different cost functions are used in the literature, usually involving a non-linearity that shapes the probability destiny function of the source signals.

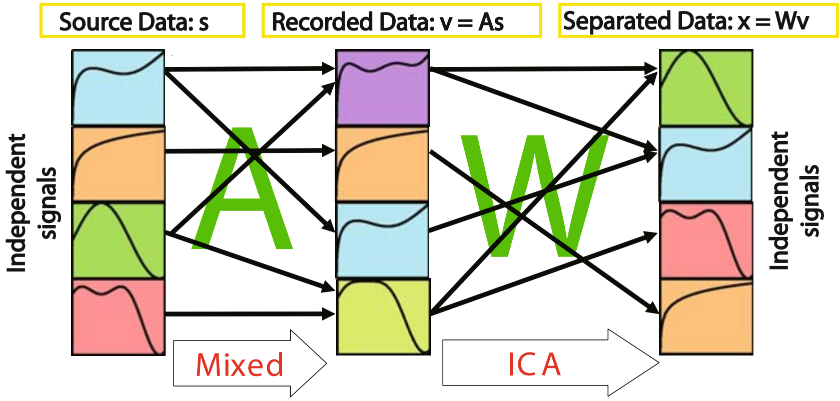


Fig. 1. Conceptual ICA algorithms

2.1 FastICA

High-order statistics, such as the kurtosis, are widely used as well. The kurtosis shows how independent the signal is because the kurtosis is the classical measure of nongaussianity [5]. The Fast ICA [5] which is one of the ICA algorithms, is based on a cost function minimization or maximization that is a function of the kurtosis ($\kappa(\mathbf{w}^\mathbf{T}\mathbf{x}) = E\{(\mathbf{w}^\mathbf{T}\mathbf{x})^4\} - 3[E\{\mathbf{w}^\mathbf{T}\mathbf{x}\}^\mathbf{2}]^\mathbf{2} = E\{(\mathbf{w}^\mathbf{T}\mathbf{x})^\mathbf{4}\} - \mathbf{3} \parallel \mathbf{w} \parallel^\mathbf{4}$; $\mathbf{w}$ is one of the rows of $\mathbf{W}$). Then the Fast ICA changes the weight $\mathbf{w}$ to extract an independent component with the fixed-point algorithm.

2.2 AMUSE

When the time-lagged covariances are all zero, each signal have mutual independence. The AMUSE uses the time-lagged covariances to calculate independence of each signals. The procedure of AMUSE algorithm is in following.

1. Whiten the (zero-mea) data $\mathbf{x}$ to obtain $\mathbf{z}(t)$.
2. Compute the eigenvalue decomposition of $\mathbf{C}_\tau^\mathbf{Z} = \frac{1}{2}[\mathbf{C}_\tau + \mathbf{C}_\tau^T]$, where $\mathbf{C}_\tau = E\{\mathbf{z}(t)\mathbf{z}(t-\tau)\}$ is the time-lagged covariance matrix, for some lag τ.
3. The rows of the separating matrix $\mathbf{W}$ are given by the eigenvectors.

2.3 NG-FICA (Natural Gradient Flexible ICA)

NG-FICA[7] uses kurtosis as an independence criterion and uses a natural (relative) gradient approach. The update functions of NG-FICA are based on the following learning formula:

$$\Delta \boldsymbol{W} = \eta \left(\boldsymbol{I} - \langle \boldsymbol{y}\boldsymbol{y}^T - \{\boldsymbol{\varphi}\boldsymbol{y}^T + \boldsymbol{y}\boldsymbol{\varphi}^T\}\rangle\right) \boldsymbol{W} \qquad (1)$$

$$\varphi_i = |y_i|^{\alpha_i - 1}\mathrm{sign}(y_i) \quad (i = 1, 2, \cdots, n) \qquad (2)$$

where η is the appropriate learning rate (constant number), $\boldsymbol{y}$ is the temporary estimated signal ($= \boldsymbol{W}\boldsymbol{x}$), and sign($y_i$) is the signum function of y_i. The Gaussian exponent α_i is derived based on the kurtosis κ_i $\left(= \left\langle {y_i}^4 \right\rangle / \left\langle {y_i}^2 \right\rangle^2 - 3\right)$ of y_i: (1) $\alpha_i = 0.8$ if $\kappa_i > 20$; (2) $\alpha_i = 1$ if $0 < \kappa_i \leq 20$; (3) $\alpha_i = 4$ if $\kappa_i \leq 0$.

2.4 JADE (Joint Approximate Diagonalization of Eigenmatrices)

One approach for estimation of ICA consists of using high-order cumulant tensor. We can use the eigenvalue decomposition of the covariance matrix to whiten the data. In another word, we transform the data so the second-order correlations are zero. As a generalization of this principle, we can use the fourth-order cumulant tensor to make the fourth-order cumulants zero, or at least as small as possible.

In order to calculate the fourth-order cumulant tensor, calculation method of the eigenvalue decomposition of the tensor is important. There are some methods to compute the eigenvalue decomposition of the tensor. One of the methods is JADE (Joint Approximate Diagonalization of Eigenmatrices).

The algorithm of JADE uses forth-order cumulant tensor and uses JADE to calculate the eigenvalue decomposition of the tensor.

3 Experimental Settings

There are two tasks in this study (See Fig. 2). The first task is to record the EEG signals during a saccade to a visual target that is his right side or left side. The second task is to record the EEG signals as a control condition when a subject dose not perform a saccade even though a stimulus has been displayed. First task and second task are called visual experiments. Each experiment has 50 trials in total: 25 on the right side and 25 on the left side.

The EEG signals are recorded through 19 electrodes (Ag-AgCl), which are placed on the subject's head in accord with the international 10-20 electrode position system. The Electrooculogram (EOG) signals are simultaneously recorded through two pairs of electrodes (Ag-AgCl) attached to the top-bottom side and right-left side of the right eye.

Recorded EEG signals are calculated by four ICA algorithms; FastICA, AMUSE, NG-FICA, JADE. In order to calculate independent components, we must clip EEG signals into specific size (We call clipping EEG signals "windowing"). Therefore, in this paper, there are 6 size windows in following.

1. Window A: -999[ms] $\sim$ 1000[ms]
2. Window B: -499[ms] $\sim$ 500[ms]
3. Window C: -349[ms] $\sim$ 350[ms]
4. Window D: -999[ms] $\sim$ 0[ms]
5. Window E: -499[ms] $\sim$ 0[ms]
6. Window F: -349[ms] $\sim$ 0[ms]

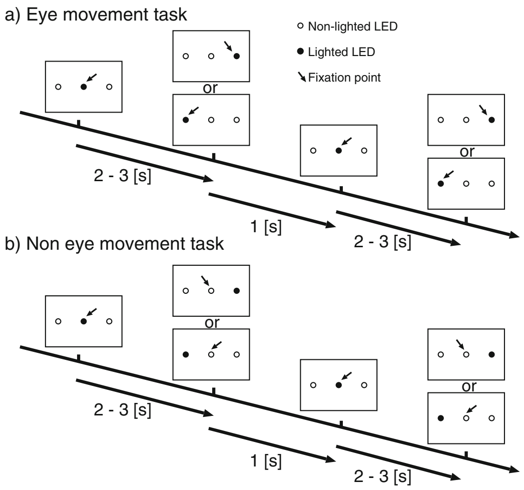


Fig. 2. Experimental Task

0[ms] indicates starting point of saccade, and minus means the time before saccade. In these windows, there are two categories; Window A ∼ C include EEG signals after saccade and window D ∼ E do not include EEG signals after saccade.

In using four algorithms and six windows, we calculate saccade-related independent components.

4 Experimental Results and Discussion

4.1 Results of FastICA

Fig.4 shows the experimental results obtained when a subject moves his eyes toward a visual target on the right side. These data are processed using the FastICA against the raw EEG data. The results shown are for Subject A, Trial #1. Top boxes represent the shapes of reference signals and bottom boxes indicate the amplitude of the ICs obtained by using the FastICA. The horizontal axes in these graphs represent the time course, where 0 [ms] indicates the start point of eye movement.

The results show that the amplitude of the signal obtained by the FastICA is sharply changed just before eye movements. The shape of the IC resembles the shape obtained with the ensemble averaging method (See Fig.3 and 4). The IC which has a peak just before eye movements bears a resemblance to the features of ensemble averaging in respect to the time when the potential incurs a sharp change. In the case of all subjects and trials, this component is extracted. Therefore, we conclude that this pre-movement component is related to the saccade-related EEG signals.

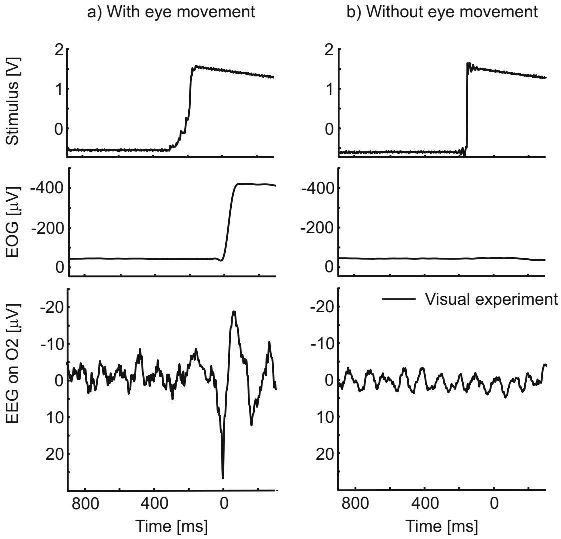


Fig. 3. Saccade-related EEG recorded on O2

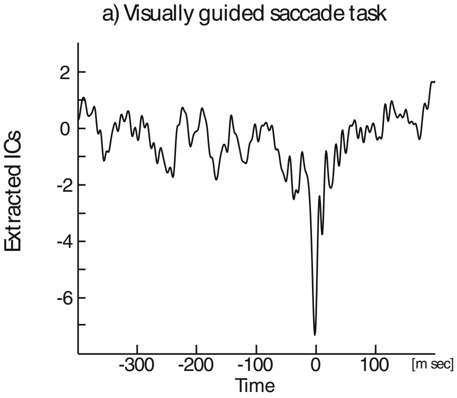


Fig. 4. Extracted signals for FastICA in visual experiments

4.2 Extraction Rate

We will determine how many the saccade-related ICs are obtained by using four ICA algorithms.

We make assumption that saccade-related IC has one big peak from -50 [ms] $\sim$ -1 [ms]. The peak-amplitude n is larger than 3; $n = \frac{\bar{x}-\mu}{s}$; where $\bar{x}$ is a mean of EEG potentials during 1000 [ms] before a saccade, μ is a maximum amplitude, and s is the standard deviation during 1000 [ms] before saccades.

Table 1. Extracted rate by four ICA algorithms

	AMUSE		FastICA		NG-FICA		JADE	
	Right	Left	Right	Left	Right	Left	Right	Left
A	24%	4%	100%	96%	29%	36%	100%	100%
B	12%	20%	84%	80%	16%	16%	92%	96%
C	36%	24%	92%	96%	8%	32%	88%	100%
D	32%	28%	96%	100%	20%	52%	96%	100%
E	20%	28%	96%	92%	28%	40%	96%	96%
Ave.	22.8%		93.2%		27.2%		96.4%	

Table 1 represents the rate for extracting saccade-related ICs from the raw EEG data by each algorithm. The extraction rate is defined by ratio:
(*the number of trials in which saccade-related IC are extracted*)
/ (The total number of trials).

In the AMUSE, the lowest rate was 4%. However, the rate for most of the subjects was over 20% and the highest rate was 36%. The average rate was 22.8%. In the FastICA, the lowest rate was 80%. However, the rate for most of the subjects was over 90% and the highest rate was 100%. The average rate was 93.2%. In the NG-FICA, the lowest rate was 8%. However, the rate for most of the subjects was over 20% and the highest rate was 40%. The average rate was 27.2%. In the JADE, the lowest rate was 88%. However, the rate for most of the subjects was over 90% and the highest rate was 100%. The average rate was 96.4%.

From these results, the FastICA and JADE yield good performances of extracting saccade-related independent components. Calculation time in Fast ICA is faster than calculation time in JADE[4]. Therefore, in this case, Fast ICA is the best in order to extract saccade-related ICs.

Next, we focused on the extracting rates in each windows (See Table 2). From Table 2, the windows not including EEG signals after saccade and the windows which has small window size had better extracting rates.

In the case of the windows including EEG signals after saccade, noise by saccade interfered with extracting of saccade-related ICs because amplitude of noise by saccade had very huge amplitude against the amplitude of another EEG signals.

Table 2. Extracted rate by six window size

	FastICA	JADE
-999 ∼ 1000 [ms]	37.2%	38%
-499 ∼ 500 [ms]	29.6%	27.2%
-349 ∼ 350 [ms]	22.4%	26.4%
-999 ∼ 0 [ms]	90%	93.6%
-499 ∼ 0 [ms]	93.2%	96.4%
-349 ∼ 0 [ms]	99.4%	99.2%

In the case of the windows which had a small window size, EEG signals cut by small windows included the small number of independent components. Therefore, it was easy to extract saccade-related ICs.

5 Conclusion

This paper presents the extraction rate of saccade-related EEG signals by four ICA algorithms and six window sizes. In terms of extracting rates focused on ICA algorithms, the JADE and Fast ICA have good results. The reason why the AMUSE and NG-FICA do not have good results is unknown. In the next step, we must check the reason why the AMUSE and NG-FICA do not have good results is unknown.

As results of the extracting rate focused on the window sizes, the window F (-349[ms] $\sim$ 0[ms]) have good results. In the case of the windows A, B, and C, EEG signals cut by small windows include the small number of independent components. Therefore, it is easy to extract saccade-related ICs in the case of small window size.

Acknowledgment

This research was supported in part by MEXT for the Grant-in-Aid for Young Scientists (B) 18700409.

References

1. Kirkup, L., Searle, A., Craig, A., Mclsaac, P., Moses, P.: EEG-based system for rapid on-off switching without prior learning. Medical and Biological Engineering and Computing 35, 504–509 (1997)
2. Wolpaw, J.R., McFarland, D.J.: Multichannel EEG-based brain-computer communication. Electroenceph. clin. Neurophysiol. 90, 444–449 (1994)
3. Funase, A., Yagi, T., Kuno, Y., Uchikawa, Y.: A study on electro-encephalo-gram (EEG) in eye movement. Studies in Applied Electromagnetics and Mechanics 18, 709–712 (2000)
4. Cichocki, A., Amari, S.: Adaptive blind signal and image processing. Wiley, Chichester (2002)
5. Hyvärinen, A., Oja, E.: A fast fixed-point algorithm for independent component analysis. Neural Computation (9), 1483–1492 (1997)
6. Tong, L., Soon, V., et al.: Indeterminacy and indentifiability of blind indentification. IEEE Trans. CAS 38, 499–509 (1991)
7. Choi, S., Cichocki, A., Amari, S.: Flexible independent component analysis. Journal of VLSI Signal Processing 26(1), 25–38 (2000)
8. Cardoso, J.-F., Souloumiac, A.: Blind beam-forming for non Gaussian signals. In: IEE Proceedings-F, vol. 140, pp. 362–370 (1993)

Policy Gradient Learning of Cooperative Interaction with a Robot Using User's Biological Signals

Tomoya Tamei and Tomohiro Shibata*

Graduate School of Information Science, Nara Institute of Science and Technology,
8916-5, Takayama, Ikoma, Nara 630-0192, Japan
{tomo-tam,tom}@is.naist.jp

Abstract. The potential market of robots that can helpfully work at home is increasing, and such robots are required to possess force and tactile sensors achieving dynamic and cooperative interactions with their users. Virtual realization of force/tactile sensors in robots, using user's biological signals such as EMG and postural information, is a versatile solution allowing high spatial resolution and degrees of freedom. In this paper, however, we first show the virtual force sensing approach does not work for a three-dimensional cooperative task in which the user is requested to move a load by an upper-limb of the user cooperatively with the robot, and discuss about inevitable problems. We then propose to apply policy gradient learning to overcome the problems, and demonstrate preliminary but promising learning results.

1 Introduction

The potential market of robots that can helpfully work at home is increasing due to, for instance, the aging population combined with a diminishing number of children. Such robots are required to possess not only visual and auditory perception but also force and tactile sensors achieving dynamic and cooperative interactions with their users. If this technology is established, the humans' affinity for welfare, livelihood-support and entertainment robots could be greatly increased. The force/tactile sensors have been usually realized by mounting force/tactile sensors over the robot's body (e.g., [1] [2]). The use of such an approach, however, is generally limited because of low spatial resolution, small degrees of freedom (DOF) of each sensor, complicated wiring, necessity of repairing, and so on. In contrast, Tamei, et al. proposed an approach whose key is the virtual realization of force/tactile sensors in robots by user's biological signals such as EMG and postural information [3]. More specifically, under a condition that a user and a robot are physically interacting, the force vector exerted by the user to the robot is estimated and transferred to the robot controller, which works as virtual force sensing. They further demonstrated a cooperative holding task in which the user is requested to move a heavy load by an upper limb of the user cooperatively with the robot vertically, armed with one-dimensional virtual force sensing perpendicular to the back of the user's hand. A natural extension of this work is to make the sensing and interaction more complicated. We then had tried three-dimensional cooperative holding task. It, however, was

* This work was supported by Grant-in-Aid for Scientific Research from Japan Society for the Promotion of Science, No. 20300071 and Mitsui Sumitomo Insurance Welfare Foundation.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 1029–1037, 2009.

found that the accuracy enough for the three-dimensional cooperative task could not be achieved by a linear function estimator, even if the enough accuracy was achieved. One possible reason could be the function approximator for virtual force sensing. In [3], a simple linear function was employed. One would argue that employing nonlinear ones could solve this problem. They, however, made sure that the performance of the linear function approximator was as good as nonlinear ones, and it was even good enough for the test dataset in the calibration procedure. Instead, there could be two inevitable problems on this matter. First, EMG patterns can be different between in the calibration stage and in the actual task, because closed-loop feedback systems can be different between in the calibration stage using real force sensing and in the actual task using virtual force sensing. Second, the muscle coordination can vary due to, for instance, fatigue. Therefore, in this study, we introduce reinforcement learning because it can make the robot controller adaptive to the changes in the EMG pattern and the muscle coordination in an on-line fashion without an explicit teacher signal, e.g., force sensor output, and conduct experiments in which a subject moves a heavy load to a specified target point cooperatively with a robot. Experimental results demonstrate the feasibility of our method.

This article is organized as follows. The next section describes our approach. Section 3 first shows a simple application of the virtual force sensing to the three-dimensional cooperative hold task does not work. Section 3.3 then describes an application of policy gradient learning to this task and demonstrates its feasibility. Section 4 concludes this study with a future direction.

2 Approach

To achieve the three-dimensional cooperative holding task, accurate estimation of the three-dimensional force vector exerted by the user's hand is required. It, however, appears inherently difficult because (1) EMG patterns can be different between in the calibration stage using real force sensing and in the actual task using virtual force sensing, and (2) the muscle coordination can vary. One could argue that some on-line learning of a function approximator would help. It, however, cannot be employed in our study, since an actual force sensor which provides a teacher signal for the learning is not allowed to use in the actual task. Therefore, we introduce reinforcement learning because it enables the robot controller to be adaptive to the changes in the EMG pattern and the muscle coordination in an on-line fashion without an explicit teacher signal. Fig. 1 illustrates our approach to applying of reinforcement learning to a cooperative interaction with a robot. The general goal of policy optimization in reinforcement learning is to optimize the policy parameters so that the expected reward is maximal. There are two conditions for applying reinforcement learning in our approach. First, a computer agent and a user should share the same goal represented as a reward function. Second, the environment should include the user; Not only the robot's state but also the user's state should be observed as much as possible. If these conditions are satisfied, the whole process would be well-approximated as an Markov decision process (MDP) which can be solved by reinforcement learning. There have been a couple of related studies that apply reinforcement learning for the interaction between a robot and a human [4] [5]. These studies, however, did not deal with or focus on physical interactions.

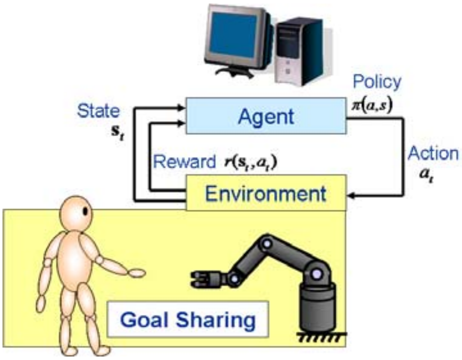


Fig. 1. Reinforcement learning of cooperative interactions with a robot

3 Experiments

In this section, we first describe experimental setup. Then, experimental results of the cooperative holding task are presented and discussed. Finally, we investigate whether policy gradient learning can overcome the problem of the task.

3.1 Experimental Setup

Fig. 2 depicts the experimental setup. As shown in this picture, a user was requested to move a heavy load using one arm and hand with a robot in a cooperative fashion. The system was comprised of a robot, a surface electromyograph and an optical motion capture device. EMG signals and markers' positional information were sent to the standard PC for controlling the robot in real time. The robot was an industrial six-DOF manipulator, PA10 (Mitsubishi Heavy Industries, Ltd.) which possessed no force sensors. EMG signals were measured by a compact-electromyograph BA1104 with active-type electrodes and the telemeter unit TU-4. Both devices are Digitex Laboratory Co., Ltd. The motion capture device was Mac3D system (Motion Analysis Corp.). The EMG signals were input to the A/D converter of Mac3D. The sampling frequency was 200 Hz for both the EMG signals and the motions. The control frequency for PA10 was also 200 Hz.

The muscles each at which EMG was recorded were Deltoid-clavicular part (DELC), Deltoid-acromial part (DELA), Deltoid-scapular part (DELS), Biceps brachii long head (BB), Pectralis major Clavicular head (PM), Triceps brachii lateral head (TB), Flexor carpi radialis (FCR) and Extensor carpi radialis longus (ECRL) (Fig. 3(a)). Six markers for motion capture were attached on the left shoulder, right side of the shoulder, elbow, wrist, and the back of the hand (Fig. 3(b)). The controlled joints of PA10 were $\theta_{\mathrm{R}1}$, $\theta_{\mathrm{R}2}$, $\theta_{\mathrm{R}3}$ and $\theta_{\mathrm{R}4}$, as shown in Fig. 4.

3.2 Cooperative Holding by Virtual Force Sensing

In this study, we first attempted to extend the work of Tamei, et al. [3] in which one-dimensional cooperative holding task was achieved by virtual force sensing. Virtual

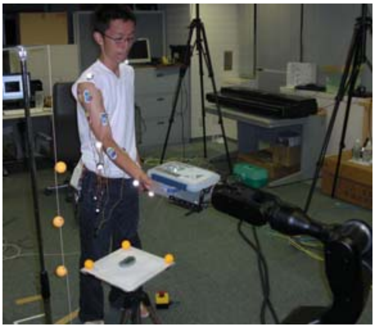
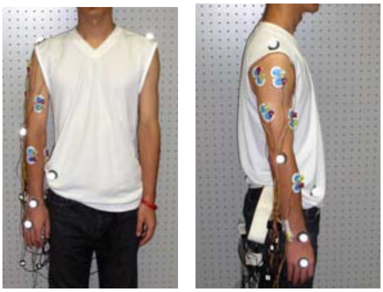


Fig. 2. Example view of the experiment



(a) Front view (b) Side view

Fig. 3. Electrodes and markers in the experiment

force sensing basically assumes the situation where a user and a robot are physically interacting, and estimates the force exerted by the user using the user's biological signals without real force sensors. We developed three-dimensional version of the cooperative holding task in which the user was asked to use only its right upper extremity. Fig. 4 is an overview of this task. For this task, we developed a three-dimensional force-vector estimator as follows. The applied force to the user's hand, $\boldsymbol{f}_{\mathrm{est}}(t) \in \boldsymbol{R}^{3\times 1}$, was linearly estimated based on EMG signals $\boldsymbol{m}(t) \in \boldsymbol{R}^{8\times 1}$ were preprocessed by full-wave rectification and low-pass filtering (cutoff frequency of 3.6 Hz), subject's joint angle vector $\boldsymbol{\theta} \in \boldsymbol{R}^{5\times 1}$ consisting of three-dimensional rotation of shoulder, on-dimensional rotation of both elbow and wrist, which were calculated from the captured motion information, as,

$$\boldsymbol{f}_{\mathrm{est}}(t)^T = [\boldsymbol{\theta}(t)^T, \dot{\boldsymbol{\theta}}(t)^T, \ddot{\boldsymbol{\theta}}(t)^T, \boldsymbol{m}(t)^T, \boldsymbol{m}(t-1)^T, \cdots, \boldsymbol{m}(t-29)^T, \boldsymbol{1}^T]^T \boldsymbol{W}. \quad (1)$$

Parameters in $\boldsymbol{W} \in \boldsymbol{R}^{256\times 3}$ were obtained by fitting to the data acquired when the subject moved the load cooperatively with the robot using a force sensor in the calibration stage. We used EMG signals with 30 tapped-delay lines corresponding to the period of 150 ms so that a suitable filter for each task would be learned. The reason is that it is known that there is a delay between an EMG signal input and the corresponding muscle contraction, and that the delay time varies to the muscle shortening velocity [6].

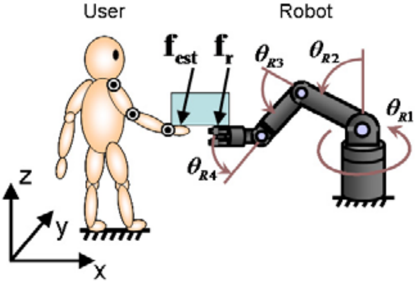


Fig. 4. Overview of the experiment

In the calibration stage, the force was measured by a force sensor and was used for controlling the robot. In one trial, the force and motion were recorded for 40 seconds, and ten trials of data were collected in total. We confirmed that the trained linear estimator was well-generalized for the data, which was a part of the collected data, through the validation test with the training and test datasets. Fig. 5 shows the performance for the test data. Root-mean-square (RMS) errors in each direction were 0.699, 0.561 and 0.750 $[\mathrm{N}]$, respectively.

The dynamics of the robot hand in the each-direction is given by

$$M_{\mathrm{R}}\ddot{\boldsymbol{p}}_{\mathrm{R}}(t) + C_{\mathrm{R}}\dot{\boldsymbol{p}}_{\mathrm{R}}(t) + K_{\mathrm{R}}\boldsymbol{p}_{\mathrm{R}} = A(\boldsymbol{f}_{\mathrm{inp}}(t) - \boldsymbol{f}_{\mathrm{u}}), \tag{2}$$

where M_{R}, C_{R}, K_{R} and A are the mass, viscosity, spring properties set to the robot hand and the amplification constant, respectively. We set these four parameters to 0.2 $[\mathrm{kg}]$, 1.0 $[\mathrm{N}\cdot\mathrm{s/m}]$, 0.1 $[\mathrm{N/m}]$, and 2.0, respectively. $\boldsymbol{p}_{\mathrm{R}} = [x_{\mathrm{R}}\ y_{\mathrm{R}}\ z_{\mathrm{R}}]^T$ is the Cartesian coordinates of the robot hand. The desired velocity of the robot hand $\hat{\dot{\boldsymbol{p}}}_{\mathrm{R}}$ was determined by the difference between the target value of the force applied to the subject's hand, $\boldsymbol{f}_{\mathrm{u}}$, and the force input to the controller, $\boldsymbol{f}_{\mathrm{inp}}(t)$. We assumed to be $\boldsymbol{f}_{\mathrm{u}} = [0\,0 - \frac{M}{2}(g+\ddot{z})]^T$ and set $M = 2$ $[\mathrm{kg}]$. g and z are the gravity acceleration and the z-coordinate of the load, respectively. The angular velocities, $\hat{\dot{\theta}}_{\mathrm{R1}}$, $\hat{\dot{\theta}}_{\mathrm{R2}}$ and $\hat{\dot{\theta}}_{\mathrm{R3}}$, sent to the robot for control, were calculated as $[\hat{\dot{\theta}}_{\mathrm{R1}}\ \hat{\dot{\theta}}_{\mathrm{R2}}\ \hat{\dot{\theta}}_{\mathrm{R3}}]^T = \boldsymbol{J}(\boldsymbol{\theta}_{\mathbf{R}})^{-1}[\hat{\dot{x}}_{\mathrm{R}}\ \hat{\dot{y}}_{\mathrm{R}}\ \hat{\dot{z}}_{\mathrm{R}}]^T$. and $\hat{\theta}_{\mathrm{R4}}$ was obtained by $\hat{\theta}_{\mathrm{R4}} = -\hat{\theta}_{\mathrm{R2}} - \hat{\theta}_{\mathrm{R3}}$, so that the robot hand be kept horizontally.

Fig. 6 presents an example result in which the experiment was conducted as $\boldsymbol{f}_{\mathrm{inp}}(t) = \boldsymbol{f}_{\mathrm{est}}(t)$. The top panel shows the changes of the estimated force applied to the subject's hand, $\boldsymbol{f}_{\mathrm{est}}(t)$. The second to the bottom panel show the trajectory of the load with the target points indicated by circles, in each direction. These figures show that the load could not be moved to the target points successfully even though the virtual force estimation was good enough for the test dataset in the calibration procedure, cf. Fig. 5.

3.3 Policy Gradient Learning of the Cooperative Holding Task

As discussed before, our desire was to achieve an accurate three-dimensional cooperation holding task. Because no explicit teacher signal could be given, this problem is formulated as a reinforcement learning problem. The overview of reinforcement learning of the task is illustrated in Fig. 7. The computer agent observes the user's EMG and motion signals and optimizes the policy $\boldsymbol{a}$ as the additional term to the estimated force. That is, the force input to the impedance controller (cf. Eq. (2),) is calculated by

$$\boldsymbol{f}_{\mathrm{inp}} = \boldsymbol{f}_{\mathrm{est}} + \boldsymbol{a}. \tag{3}$$

As stated in Section 2, the user and the robot should share the same goal. For this, the task was redefined as a reaching task, i.e., the user and the robot was requested to move a load cooperatively to a specified location in specified time. That is, the spatiotemporal specification was the shared goal.

We employed the on-line version of GARB algorithm [7] for reinforcement learning. The on-line GARB is a policy-gradient method which is suitable for robot learning problem. The advantages of the policy gradient method are that the policy representation can be chosen so that it is meaningful for the task and can incorporate domain

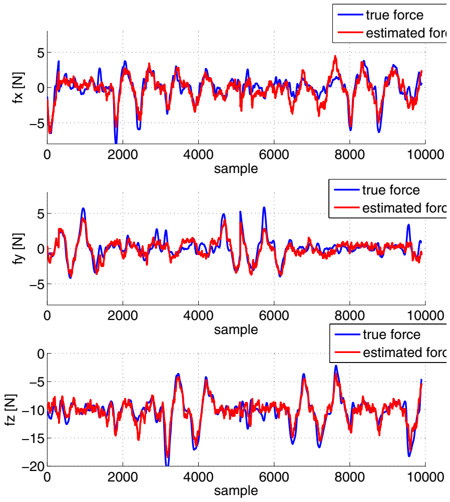


Fig. 5. Validation of the linear estimator

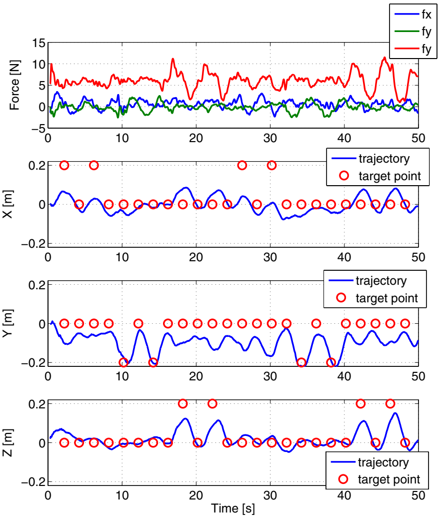


Fig. 6. Results of the 3D cooperative holding task

knowledge, that often fewer parameters are required in the learning process than in value-function based methods. In addition, the policy gradient methods can be used model-free.

The state $\boldsymbol{s} \in \boldsymbol{R}^{8\times1}$, policy $\boldsymbol{a} \in \boldsymbol{R}^{3\times1}$ and reward $r \in \boldsymbol{R}$ were defined as

$$\boldsymbol{s} \equiv (\boldsymbol{m}, \boldsymbol{p}_{\mathbf{R}}, \boldsymbol{\theta}_{\mathbf{R}}), \tag{4}$$

$$a_i \sim \pi(a_i|\boldsymbol{s}, \boldsymbol{w}_i) = \mathcal{N}(a_i|\mu_i, \sigma_i), \tag{5}$$

$$\mu_i = \sum_{j}^{N_{\text{EMG}}} w_{ij} m_j,$$

$$\sigma_i = \frac{1}{1 + \exp -w_{i9}}, \quad (i = x, y, z),$$

$$r = \frac{1}{(2\pi)^2 \left|\boldsymbol{\Sigma}\right|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{d})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{d})\right\} \tag{6}$$

$$-\gamma_{\text{A}} \sum_{k}^{N_{\theta_{\text{R}}}} \exp\left(-\ddot{\theta}_{\text{R}k}^{\,2}\right) - \gamma_{\text{EMG}} \sum_{j}^{N_{\text{EMG}}} m_j, \tag{7}$$

where $\boldsymbol{w}_i$ is the parameter vector which were learned by reinforcement learning. $N_{\theta_{\text{R}}}(= 4)$ and $N_{\text{EMG}}(= 8)$ are the number of controlled joint of the robot and the measured EMG signals, respectively. The reward function consists of three terms; The first term represents the reaching accuracy in time and space, the second term for smoothness in robot motion, and the third term for energy efficiency. The first term was a normal probability density function in which $\boldsymbol{d}$ is the desired load state specified in time t_{target} [s] and the target position in the world coordinates (x_{target}, y_{target} and z_{target} [m]).

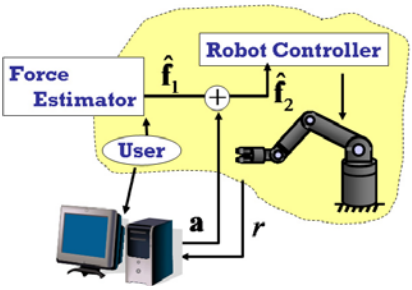
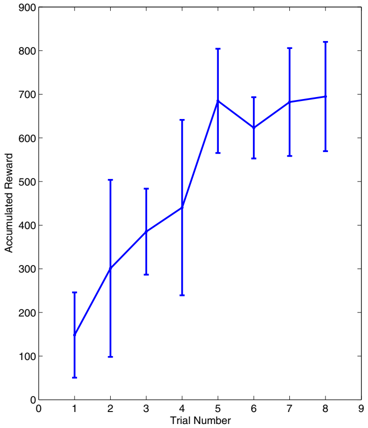


Fig. 7. Implementation of reinforcement learning in the experiment



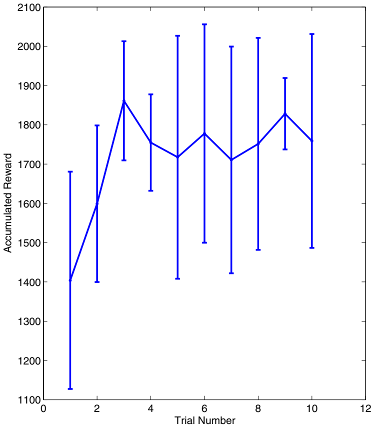


Fig. 8. Learning curve obtained in task 1

Fig. 9. Learning curve obtained in task 2

$\boldsymbol{x}$ is the current load state. $\boldsymbol{\Sigma}$ is a diagonal matrix whose elements were set to 0.2 and 0.002. m_j is the EMG signal of j-th muscle. γ_{A} and γ_{EMG} are constants balancing the contribution of the three terms, and both were empirically set to 0.1.

We selected the current EMG signals $\boldsymbol{m}$ for the agent's state $\boldsymbol{s}$, based on an offline analysis; the EMG signals had the largest coefficient of determination responsible for the force estimation error with linear approximation function, compared to other signals such as the joint motion and hand tip motion of the subject. The parameters α_p and β for GARB were configured to 10^{-6} and 0.99, cf., Appendix.

We conducted two tasks seven times (seven episodes). The desired load state $\boldsymbol{d}(= [t_{\text{target}}\ x_{\text{target}}\ y_{\text{target}}\ z_{\text{target}}]^T)$ is $[2.0\ 0.2\ 0.0\ 0.0]^T$ and $[2.0\ 0.0\ 0.0\ 0.2]^T$ in task 1 and task 2, respectively. In each trial, the task was terminated in 2.5 $[\mathrm{s}]$.

Fig. 8 and 9 show learning curves, i.e., mean and standard deviation of the accumulated reward over seven episodes, obtained in task 1 and 2, respectively. As shown in these figures, policy gradient learning of each cooperative holding task was quickly accomplished.

4 Conclusion

In this study, a cooperative holding task was first attempted based on the virtual force sensing approach [3]. The force estimation was achieved by linear function approximation from the motion and EMG signals of the user's upper-limb. Although it was found that this simple linear estimation worked well, even for test data, in the calibration phase, we found that it did not work in the actual task, probably due to two major reasons: (1) EMG patterns were different between in the calibration stage and in the actual task, (2) the muscle coordination varied, e.g., it is known that many muscles contribute to shoulder motion, and that torque vector of each muscle changes dramatically according to the shoulder posture [8] [9]. We then introduced a policy gradient method, which is a reinforcement learning method, as it has a possibility to make the robot controller cope with the changes in the EMG pattern and the muscle coordination in an on-line fashion without an explicit teacher signal. The goal shared by the user and the computer agent as a reward function was quickly and stably achieved through learning experiments.

Note that, as stated in section 2, our approach is not confined to the cooperative holding task. Application of our approach to other tasks such as motor learning and rehabilitation is our future work.

References

1. Miyashita, T., Tajika, T., Ishiguro, H., Kogure, K., Hagita, N.: Haptic communication between humans and robots. In: Proc. 12th Int. Symp. Robot Res. (2005) (CD-ROM)
2. Hoshi, T., Shinoda, H.: A tactile sensing element for a whole body robot skin. In: Proc. 12th Int. Symp. Robot Res. (2005) (CD-ROM)
3. Tamei, T., Ishii, S., Shibata, T.: Virtual force/tactile sensors for interactive machines using the user's biological signals. Adv. Robot. 22(8), 893–911 (2008)
4. Tapus, A., Tapus, C., Mataric, M.J.: Hands-off therapist robot behavior adaptation to user personality for post-stroke rehabilitation therapy. In: Proc. IEEE Int. Conf. Robot Automat., pp. 1547–1553 (2007)
5. Mitsunaga, N., Smith, C., Kanda, T., Ishiguro, H., Hagita, N.: Adapting robot behavior for human-robot interaction. IEEE Trans. Robot 24(4), 911–916 (2008)
6. Corcos, D.M., Gottlieb, G.L., Latash, M.L., Almeida, G.L., Agarwal, G.C.: Electromechanical delay: An experimental artifact. J. Electromyogr. Kinesiol. 2, 59–68 (1992)
7. Weaver, L., Tao, N.: The optimal reward baseline for gradient-based reinforcement learning. In: Proc. 17th Conf. Uncertainty in Artificial Intelligence, pp. 538–545 (2001)
8. Yoshida, N., Domen, K., Koike, Y., Kawato, M.: A method for estimating torque-vector directions of shoulder muscles using surface EMGs. Bio. Cybern. 86(3), 167–177 (2002)
9. Buneo, C.A., Soechting, J.F., Flanders, M.: Postural dependence of muscle actions: Implications for neural control. J. Neurosci. 17(6), 2128–2142 (1997)

Appendix: GPOMDP Algorithm

(1) The agent observed the state $\boldsymbol{s_t}$ from the environment and take action $\boldsymbol{a_t}$ based on probabilistic policy $\pi(\boldsymbol{a_t}, \boldsymbol{s_t}, \boldsymbol{w(t)})$,

(2) is given reward r_t and observed the next state $\boldsymbol{s_{t+1}}$.
(3) update action selection probability

$$\begin{aligned} e_i(t) &= \frac{\partial}{\partial w_i(t)} \ln\left(\pi(\boldsymbol{a_t}, \boldsymbol{s_t}, \boldsymbol{w}(t))\right) \\ D_i(t) &= e_i(t) + \beta D_i(t) \\ b(t) &= b(t-1) + \frac{1}{t}(r(t) - b(t-1)) \\ \Delta w_i(t) &= (r(t) - b(t)) \\ \boldsymbol{w}(t+1) &= \boldsymbol{w(t)} + \alpha_p \Delta \boldsymbol{w}(t) \end{aligned}$$

$e_i(t)$ and $D_i(t)$ are eligibility and eligibility trace. $\boldsymbol{w}(t) = (w_1(t), w_2(t), \cdots, w_l(t))$ is parameters. β $(0 < \beta < 1)$ and α_p are discount ratio of eligibility trace and learning ratio, respectively.
(4) set a time t forward and return to step (1).

Real-Time Embedded EEG-Based Brain-Computer Interface

Li-Wei Ko[1,2], I-Ling Tsai[1,2], Fu-Shu Yang[1], Jen-Feng Chung[1,2],
Shao-Wei Lu[1], Tzyy-Ping Jung[1,3], and Chin-Teng Lin[1,2]

[1] Brain Research Center, National Chiao Tung University, 1001 University Road, Hsinchu 30010, Taiwan, R.O.C.
[2] Department of Electrical and Control Engineering, National Chiao Tung University, 1001 University Road, Hsinchu 30010, Taiwan, R.O.C.
[3] Swartz Center for Computational Neuroscience, University of California San Diego, Suite 320, 4150 Regents Park Row, La Jolla CA 92037-1417, USA
lwko@mail.nctu.edu.tw, sandy.iling@gmail.com,
yfs0606@hotmail.com, jfchung71@gmail.com,
swlucifer@mail.nctu.edu.tw, jung@sccn.ucsd.edu,
ctlin@mail.nctu.edu.tw

Abstract. Online artifact rejection, feature extraction, and pattern recognition are essential to advance the Brain Computer Interface (BCI) technology so as to be practical for real-world applications. The goals of BCI system should be a small size, rugged, lightweight, and have low power consumption to meet the requirements of wearability, portability, and durability. This study proposes and implements a moving-windowed Independent Component Analysis (ICA) on a battery-powered, miniature, embedded BCI. This study also tests the embedded BCI on simulated and real EEG signals. Experimental results indicated that the efficacy of the online ICA decomposition is comparable with that of the offline version of the same algorithm, suggesting the feasibility of ICA for online analysis of EEG in a BCI. To demonstrate the feasibility of the wearable embedded BCI, this study also implements an online spectral analysis to the resultant component activations to continuously estimate subject's task performance in near real time.

1 Introduction

Brain Computer Interface (BCI) technology is a research field that has emerged and grown rapidly over the past 15 years [1-2]. The EEG-based BCI system comprises a set of EEG sensors and signal processing components that acquire and analyze brain activities to directly establish a reliable communication channel between the brain and an external device. One of the biggest challenges in an EEG-based BCI system lies on the contaminations from pervasive EEG artifacts from eye movements, blinks, muscle, heart, and line noise. Independent Component Analysis (ICA) [3] is a novel statistical technique that aims at finding linear projections of the data that maximize their mutual independence. ICA has been proven to be an effective technique to remove various types of artifacts [4]. ICA has also been used to extract informative features from the data [1][4-5]. Hill et. al. [7-9], for example, demonstrated the use of ICA in

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 1038–1045, 2009.

an EEG-based BCI. However, ICA and other signal-processing functions were performed offline on a personal computer in most of these studies, which hinders the wearability, portability and practical use of the systems in operational environments. Given the recent development of embedded system and signal processing techniques, it is now practical to implement these sophisticated algorithms in real-time embedded systems for on-line EEG monitoring and/or BCI. This study details the design and testing of a near real-time embedded BCI featuring online ICA and spectral analysis for continuously monitoring cognitive states of participants performing realistic driving tasks in a virtual reality-based dynamic driving environment.

2 Materials and Methods

2.1 Simulation Dataset

To test and optimize the online ICA, we first generated simulation dataset by summing three super-Gaussian and a uniformly distributed white noise signals.

2.2 Subjects and Driving Task Experiments

Four subjects participated in a Virtual-Reality (VR)-based highway driving experiment, in which they were instructed to put forth their best effort to keep their lane position. An actual automobile was mounted on a 6- DOF Stewart platform, which provides translational and rotational movement and vibratory feedback to simulate actual driving conditions. The 360o projection of driving scenery is updated synchronously with deviations caused by wheel/paddle movement by the subjects or by road conditions. Every 3-7 seconds the car was linearly pulled towards the curb or into the opposite lane, with equal probability. Subjects were instructed to compensate for the drift by steering a vehicle wheel. The EEG data were recorded at Fp1, Fpz, Fp2 and midway between Fpz and nasion, referentially against a right-mastoid reference by a Neuroscan amplifier. The EEG data were sent to a PC or a miniature embedded BCI for further analysis. Driving performance was measured by the distance of lane deviation, which was small when the subject was alert, and vice versa. The driving parameters (lane position and wheel rotation) were in sync with the EEG acquisition system.

2.3 Independent Component Analysis

Given a *N*-dimension data vector $X(t)=[\chi_1(t), \ldots, \chi_N(t)]^T$ observed at each time point, the goal of ICA is to find a linear unmixing matrix such that the unmixed components, $Y(t)$ $[y_1(t), \ldots, y_N(t)]^T$ of the linear transform, $Y(t)=WX(t)$, of are statistically independent. The ICA methods were extensively applied to blind source separation problem since 1990s [3].

Figure 1 shows the flowchart of ICA. The observed data matrix was first processed by “centering” to remove the means of each variable (rows of the data matrix). After centering, whitening transformation was used to make the zero-mean time series uncorrelated and having equal variances. The resultant data were then decomposed by “infomax” ICA [10].

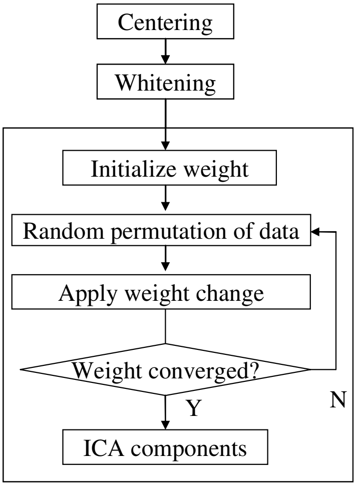


Fig. 1. The flowchart of the "infomax" ICA algorithm

2.4 Online Independent Component Analysis Implementation

ICA was successively applied to EEG data of 5-sec (320-points) windows with a 3-sec (192-point) overlap. Figure 2 depicts the differences between offline and online ICA methods. To ensure the near real-time ICA processing, we limited the numbers iteration and the convergence tolerance of training weights in the infomax algorithm. To test the effects of these parameters on training time and performance, we stopped the ICA training when the weights changes are less than, 0.00002 (setting 1), 0.0002 (setting 2), and 0.002 (setting 3).

2.5 Moving-Average Power Spectral Analysis

Moving-averaged spectral analysis of the EEG data was accomplished using a 64-point Hanning-window with 25% overlap. Windowed 32-point epochs were extended to 64 points by zero-padding. Median filtering using a moving 3-s (192-pt) window was used to further minimize the presence of artifacts in the EEG records.

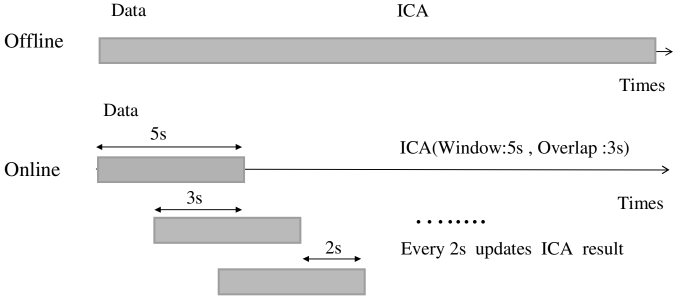


Fig. 2. Comparison of offline and online ICA implementations

The moving-averaged EEG power spectra were further converted to a logarithmic scale prior to further analysis. Logarithmic scaling linearizes the expected multiplicative effects of sub-cortical systems involved in wake-sleep regulation on EEG amplitude [11]. Several studies have reported that EEG power spectra at theta (4~7Hz) band [12-13], and/or alpha power (8~12) band [14-15] were associated with human drowsiness, and the relationship between EEG log power and subject task performance was largely linear. We thus used the resultant alpha and theta log power time series at 2-sec step in this study. Figure 3 illustrates the spectral analysis.

The time series of recorded behavior data (driving performance measured by the distance of lane deviation) and alpha & theta log spectra were smoothed using a causal 90-s square moving-averaged filter advancing at 2-sec steps to eliminate variance at cycle lengths shorter than 1-2 minutes since driving performance becomes erratic at cycle lengths of 4 minutes and longer [12-13]. The correlation coefficients between the smoothed subjects' driving error and the subband log power spectra of all ICA components at each frequency band were calculated to assess the relationship between subject task performance and EEG log spectra.

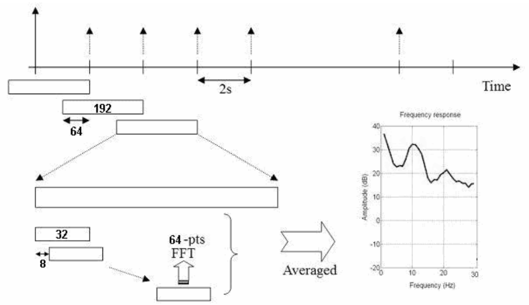


Fig. 3. Moving Average Power Spectrum for ICA components

2.6 Embedded Brain Computer Interface System

To be practical for operational environments, the BCI must be light-weight, portable, wearable and battery-powered. We developed a portable and wearable embedded BCI platform (as shown in Figure 4) accommodating the online ICA and signal processing methods mentioned above. The core processor selected for this BCI system platform is ADSP-BF533 by Analog Devices Inc. The maximum high performance of BF533 processor can reach 600MHz. It has two 16-bit MACs, two 40-bit ALUs, four 8-bit video ALUs, and 40-bit shifter. Besides the Blackfin processor, the embedded BCI system platform includes:

- 16 MB SDRAM (64M x 16 bits) and 4 MB FLASH memory
- RS-232 serial interface
- 6 keypads and 240 by 320 pixels LCD
- JTAG interface for debug and FLASH programming
- Bluetooth transceiver module

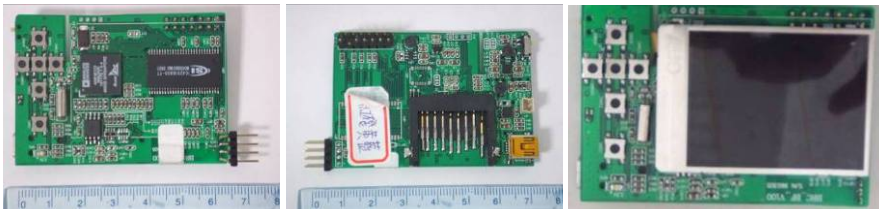


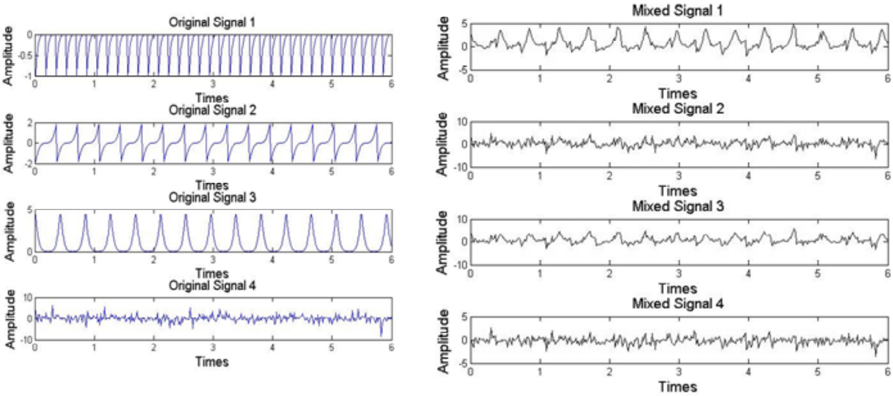
Fig. 4. The Picture of DSP Platform (Dimension: 64 x 45 mm^2)

3 Experimental Results

3.1 Experimental Results in Simulation Dataset

Figure 5 shows the original simulated source and mixture signals. Figures 6-8 show the results of the offline ICA, online ICA on a PC and online ICA on the embedded DSP platform, respectively. The resultant component activations from 3 implementations were very comparable (Pearson correlations all above 0.87).

Table 1 shows the effects of ICA training stopping parameter (weight changes) on the convergence iteration and time. Figure 9 shows the resultant component activations an obtained with different settings (0.00002, 0.0002, and 0.002). As can be seen, the results are very comparable (Setting 1 vs 2, r = 0.87-0.995; Setting 1 vs 3, r = 0.94 – 0.98).



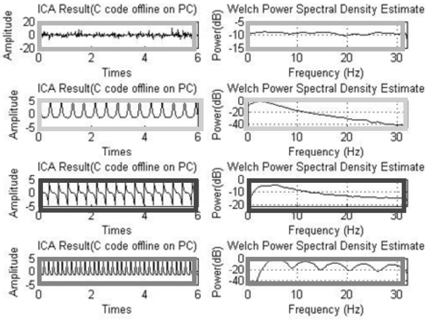
(a)

(b)

Fig. 5. (a) The simulated sources of Gaussian signals, (b) random mixing signals

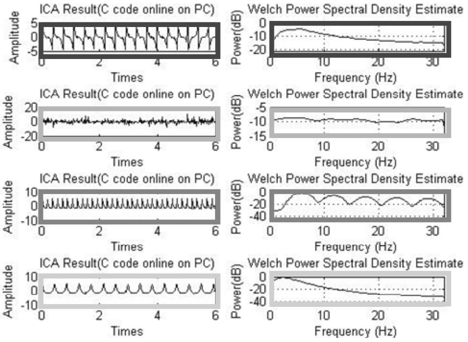
Table 1. The Experimental Results of Online ICA

Online ICA weight change in 1 min data		Setting 1 0.00002	Setting 2 0.0002	Setting 3 0.002
Simulated Signals	Average Process Time	10.26s	3.78s	0.7s
	Average Iteration Cycles	47	18.7	3
Real EEG Signals	Average Process Time	8.78s	3.29s	0.77s
	Average Iteration Cycles	48.86	17.89	3.64



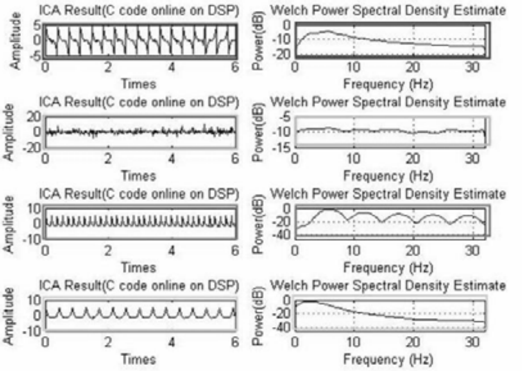
(a) (b)

Fig. 6. Experimental Results of Offline ICA Implementation in a PC, (a) Four Sources of ICA Components, (b) Power Spectra of the Component Activations



(a) (b)

Fig. 7. Experimental Results of Online ICA Implementation in a PC, (a) Four Sources of ICA Components, (b) Power Spectra of the Component Activations

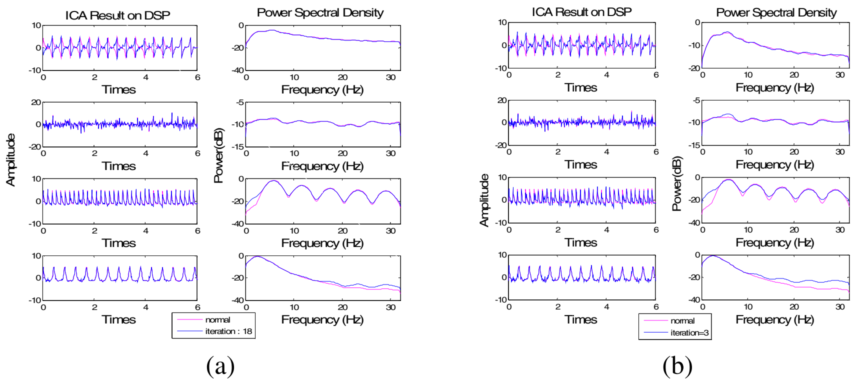


(a) (b)

Fig. 8. Experimental Results of Online ICA Implementation in the DSP Platform, (a) Four Sources of ICA Components, (b) Power Spectra of the Component Activations

3.2 Subject Performance Estimation by Using the Embedded BCI

In this study, ICA and moving spectral analysis implemented on the wearable embedded BCI were continuously applied to the ongoing EEG while the participants were performing lane-keeping driving tasks. Table II shows the correlation coefficients between the smoothed time series of and subject driving performance and the theta and alpha power of most task-performance-related components. The correlations, in general, were very high, consistent with our previous results [11] using a PC as a BCI platform.



(a) (b)

Fig. 9. (a) Four component activations in setting 2 over-plotted in setting 1 (red). (b) Four component activations in setting 3 over-plotted in setting 1 (red).

Table 2. The Correlation Results of Four Subjects

Subject		S1	S2	S3	S4
Offline	Theta	0.7316	0.8978	0.1835	0.8506
	Alpha	0.019	0.8686	0.8271	0.3693
Online	Theta	0.8395	0.7574	0.4728	0.6952
	Alpha	0.6848	0.7527	0.7823	0.4621

4 Discussions and Conclusions

The unavailability of a BCI capable of online signal processing and artifact correction or separation has long limited the use of BCI in operational environments. This study implemented a moving-windowed online ICA and spectral estimation on a miniaturized, battery-powered and light-weight embedded BCI. Our empirical results showed that the efficacy of online signal separation was comparable to that of the offline implementation. The remaining issue is to develop an algorithm to automatically select the performance-related independent component(s).

In conclusion, this study demonstrated the feasibility of online signal processing and source separation on a wearable miniature embedded BCI. This demonstration could lead to a practical wearable BCI for the monitoring of the brain functions of unconstrained participants performing normal tasks in the workplace and home.

Acknowledgments. This work was supported by the National Science Council, Taiwan, R.O.C., under Contracts NSC 97-2627-E-009-001 and NSC NSC 97-2220-E-009-052; the "Aiming for the Top University Plan," National Chiao-Tung University, the Ministry of Education, Taiwan, under Contract 97W806; and U.S. Army Research Laboratory under Grant W911NF-08-2-0003. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army, or the

U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation heron.

References

1. Kachenoura, A., Albera, L., Senhadji, L., Comon, P.: ICA: A Potential Tool for BCI Systems. IEEE Signal Processing Magazine 25(1), 57–68 (2008)
2. Kalcher, J., Flotzinger, D., Neuper, C., Gölly, S., Pfurtscheller, G.: Graz brain–computer interface II: Toward communication between humans and computers based on online classification of three different EEG patterns. Med. Biol. Eng. Comput. 34, 382–388 (1996)
3. Comon, P.: Independent component analysis, a new concept? Signal Process 36(3), 287–314 (1994)
4. Jung, T.P., Humphries, C., Lee, T.W., Makeig, S., McKeown, M.J., Iragui, V., Sejnowski, T.J.: Extended ICA removes artifacts from electroencephalographic recordings. Advances in Neural Information Process Systems 10, 894–900 (1998)
5. Vigário, R., Särelä, J., Jousmäki, V., Hämäläinen, M., Oja, E.: Independent Component Approach to the Analysis of EEG and MEG Recordings. IEEE Transactions on Biomedical Engineering 47(5), 589–593 (2000)
6. Jung, T.P., Makeig, S., Lee, T.W., McKeown, M.J., Brown, G., Bell, A.J., Sejnowski, T.J.: Independent component analysis of biomedical signals. In: Proceedings of the 2nd International Workshop on Independent Component Analysis and Blind Signal Separation, pp. 633–644 (2000)
7. Hill, N.J., Lal, T.N., Bierig, K., Birbaumer, N., Scholkopf, B.: Attentional modulation of auditory event-related potentials in a brain-computer interface. In: Proc. IEEE Int. Workshop Biomedical Circuits and Systems, Singapore, pp. 17–20 (2004)
8. Kamousi, B., Liu, Z., He, B.: Classification of Motor Imagery Tasks for Brain-Computer Interface Applications by Means of Two Equivalent Dipoles Analysis. IEEE Trans. on Neural Systems and Rehabilitation Engineering 13(2), 166–171 (2005)
9. James, C.J., Wang, S.: Blind source separation in single-channel EEG analysis: An application to BCI. In: Proc. 28th Annu. Int.Conf. IEEE Engineering in Medicine and Biology Society, New York, USA, August 2006, pp. 6544–6547 (2006)
10. Lee, T.W., Girolami, M., Sejnowski, T.J.: Independent component analysis using an extended infomax algorithm for mixed sub-Gaussian and super-Gaussian sources. Neural Computation 11, 606–633 (1999)
11. Lin, C.T., Wu, R.C., Liang, S.F., Chao, W.H., Chen, Y.J., Jung, T.P.: EEG-based drowsiness estimation for safety driving using independent component analysis. IEEE Transactions on Circuits and Systems I 52(12), 2726–2738 (2005)
12. Makeig, S., Jung, T.P.: Tonic, phasic, and transient EEG correlates of auditory awareness in drowsiness. Cognitive Brain Research 4, 15–25 (1996)
13. Makeig, S., Jung, T.P., Sejnowski, T.J.: Awareness during drowsiness: dynamics and electrophysiological correlates. Canadian Journal of Experimental Psychology 54, 266–273 (2000)
14. Schier, M.A.: Changes in EEG alpha power during simulated driving: a demonstration. International Journal of Psychophysiology 37, 155–162 (2000)
15. Joutsiniemi, S.L., Kaski, S., AndreoLarsen, T.: Self-organizing map in recognition of topographic patterns of EEG spectra. IEEE Transactions on Biomedical engineering 42(11), 1062–1068 (1995)

Part IX

Neural Network Implementations

SpiNNaker: The Design Automation Problem

Andrew Brown[1], David Lester[2], Luis Plana[2], Steve Furber[2], and Peter Wilson[1]

[1] Electronics and Computer Science, University of Southampton, UK
{adb,prw}@ecs.soton.ac.uk
[2] Computer Science, University of Manchester, UK
{drl,lap,sbf}@cs.man.ac.uk

Abstract. This paper describes the design automation issues and techniques used to design a massively parallel processing platform – SpiNNaker – from a hardware and systems design perspective. The emphasis of this paper is addressing the key problem of resource mapping, where multiple threaded programs are to be targeted onto a hardware platform that consists of multiple ARM cores and other resources such as memory and networks. In addition, the design environment is considered to ensure that a designer can program applications onto this environment in a practical manner.

1 Introduction

SpiNNaker is a massively parallel multi-core computing engine, consisting of a vast array of ARM cores and a fast interconnect fabric. Although strictly a clocked system, each ARM core is effectively decoupled from its peers, and individual processors communicate with each other by means of packets; a packet incident upon a processor causes an interrupt which handles the packet. The system has been designed to support the real-time simulation of large aggregates of spiking neurones. The development strategy is coarsely incremental, but the final goal is to be able to simulate aggregates of a billion neurones, where each of a million processors is supporting the emulation of a thousand neurones.

Various aspects of the system (physical architecture and interconnect fabric, neural and synaptic models) have been described in detail elsewhere 1.2.3. The overall structure of the system is shown in figure 1. This paper describes some of the problems associated with mapping the abstract neural connection topology (which may, but is not required to be, three dimensional) onto a physical two dimensional array of processors. The requirements are highly analogous to the automated place and route (APR) problem experienced in chip design, and the evolution of those problems almost identical. In the world of IC design, components representing the realisation of a circuit (transistors, resistors, vias and so on) have to be laid out on a two dimensional silicon die, and then the geometry of the interconnect defined. In the early days of IC design, this could be done by hand, but as the size and complexity of systems grew, design automation - initially a luxury - became a necessity. Today, it is simply not possible to lay out a state-of-the-art IC (5mm x 5mm die, feature size O(100nm)) by hand. So it is with SpiNNaker: the neural systems we wish to simulate are vast topologies of interconnected neurones, which have to be mapped onto our array of processors. The situation is actually worse than in the electronic counterpart: in

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 1049–1056, 2009.

electronic designs, design automation tools and techniques capitalise heavily on a hierarchical input description, an advantage that is largely denied us in the current problem domain. This paper describes the development strategy for a suite of APR tools designed to be used to load the SpiNNaker data structures with large ($O(10^9)$) interconnected neurones.

2 The Hardware Platform

2.1 The Chip

The internal architecture of the SpiNNaker chip is depicted in figure 1. Each chip contains 20 ARM processors (with a small amount of local memory in a Harvard configuration) and 6 bi-directional inter-chip link ports. These are interconnected by an on-chip network. External interfaces are also provided to a single bank of chip-local DRAM and Ethernet.

The chips themselves are connected (via the inter-chip link ports) in a hexagonal mesh, mapped onto the surface of a toroid. This conveniently avoids edge effects, although it requires that the notion of geometric hop distance be handled rather carefully. However, this particular design decision is one of the more easily overturned: The economics of getting silicon right first time are significantly different to changing the inter-chip layout on a PCB.

Neurones are mapped statically onto an individual ARM processor, and the internal state of the neurones mapped onto each die is held in the associated DRAM. When a spike arrives at a processor, it fetches the state of the relevant neurone from the DRAM, processes the incoming spike, updates the state of the neurone in the DRAM and may broadcast spikes of its own.

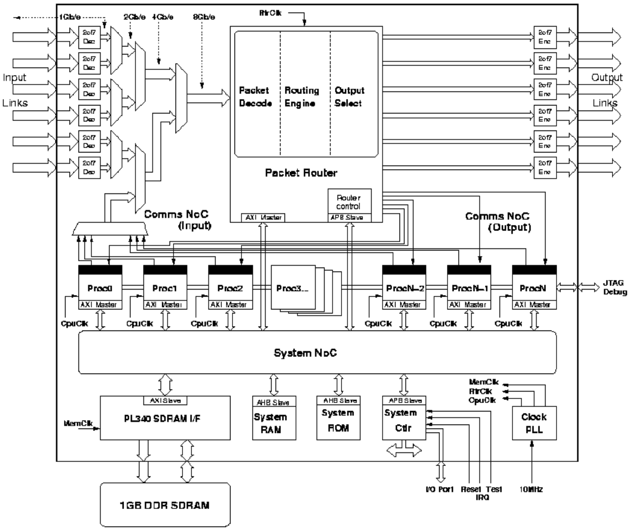


Fig. 1. A single SpiNNaker node internals

3 Place and Route

3.1 A Traditional Methodology

The APR problem is conventionally broken down into the sub-problems of *placement*, *global* and *detailed* routing.

3.1.1 Placement

Placement (in the context of SpiNNaker) involves choosing a mapping between the neurones of the abstract topological circuit and the fixed geometry of the processor array. This placement is only weakly influenced by the properties of the interconnect.

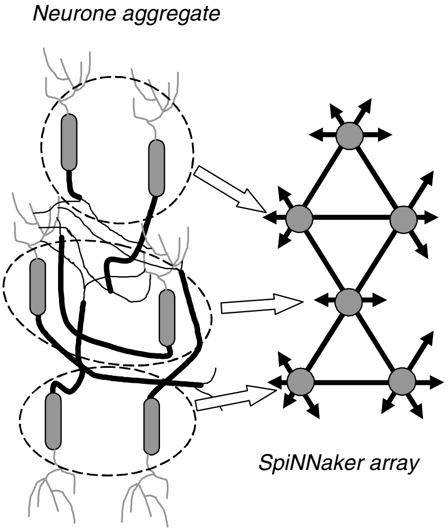


Fig. 2. Fascicle mapping

3.1.2 Routing

The routing problem is concerned with finding a route *between* a set of points, *round* a set of obstructions (placed modules) on a two-dimensional plane. The routing problem is generally decomposed into *global* and *detailed* routing, the difference being the granularity of the analysis.

Routing algorithms are even more diverse and numerous than placement, and offer the usual spectrum of reliability and quality-of-solution vs. speed. One of the earliest - and possibly most versatile - is a graph-searching algorithm, known as Lees' maze-runner 6. The algorithm can be used at arbitrary levels of granularity, and is guaranteed to find a solution for a given *single* route if it exists. On the other hand, it is relatively slow, and the overall success in finding a solution for a *set* of routes is not guaranteed. Numerous enhancements to reduce memory footprint and runtime exist, but the algorithm in its simplest form is widely applicable to all manner of graph-searching problems.

3.2 SpiNNaker-Specific Issues

The APR problem in the context of the SpiNNaker system may be summarized by figure 2: a fragment of circuitry, comprising six neurones, is to be mapped onto the fixed array of processing nodes. Each node can accommodate around 1000 neurones. Having decided upon this mapping, the routes (the sequences of nodes between source and target) must be established, and the routing tables defined in each node. What is different about the SpiNNaker context?

3.2.1 Data Size

Conventional processing speeds have increased by many orders of magnitude over the past few decades, but even this is not enough to overcome polynomial complexity in an algorithm when the input datasets become large. APR of electronic circuits containing a billion components is feasible today in reasonable timescales, *but* these circuit descriptions are highly hierarchical, the decomposition being determined and fixed by human input.

The discussion of where we actually get (meaningful) circuits of a billion neurones has yet to be published, but irrespective of whether these circuits are generated semi-automatically or stochastically generated, the APR task will be formidable.

- 32 bit machines can only address four billion memory locations; it will not be possible to even hold (let alone process) the entire datastructure at one time in an APR machine.
- Even O(n^3) algorithms - generally considered to be acceptable for APR problems - will be unusable.

The unavoidable outcome of these points is that aggressive hierarchical decomposition of the neural aggregate descriptions is an absolute necessity.

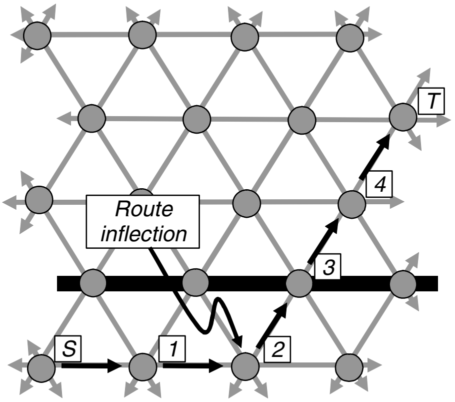


Fig. 3. Chip level SpiNNaker interconnect

3.2.2 Chip Level Topology

The SpiNNaker chips have been designed with six bidirectional I/O ports each, lending themselves naturally to a hexagonal placement on a two-dimensional plane. Six links were chosen to support a measure of fault tolerance in the final system, by providing a simple triangular bypass to each link if one fails. Identifying opposing edges of the array of chips folds the system naturally onto a toroid.

Point-to-point packet routing only requires routing table entries in certain nodes. Consider figure 3: a packet is to be sent from node S to node T; the route has been chosen to be S-1-2-3-4-T. The only nodes that require a routing table entry are S (this is *from* me), 2 (turn a corner) and T (this is *for* me). Packets incident on nodes with no corresponding routing table entry are simply passed through in a 'straight line' - the default route.

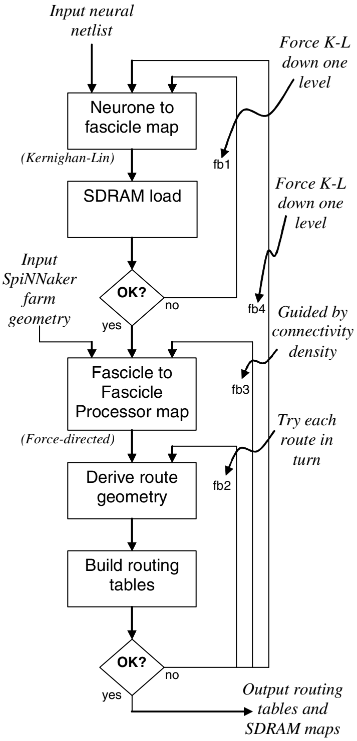

Input neural netlist
Neurone to fascicle map
(Kernighan-Lin)
SDRAM load
Input SpiNNaker farm geometry
OK?
no
yes
Fascicle to Fascicle Processor map
(Force-directed)
Derive route geometry
Build routing tables
OK?
no
yes
Output routing tables and SDRAM maps
Force K-L down one level
fb1
Force K-L down one level
fb4
Guided by connectivity density
fb3
Try each route in turn
fb2

Fig. 4. APR heuristic

It is necessary to define some terminology before going further: A **fascicle** is the collective noun for a set of neurones. Other than to note that no neurone can be a member of more than one fascicle, the term *defines* nothing about any connectivity. There is an implication (and an efficiency assumption) that the neurones in a fascicle share common input fascicles (probably sparsely connected) and common output target fascicles (again, probably sparsely connected). There is no implication that they are interconnected with each other, though they can be (or not). However, none of this is *required* - it just makes the datastructure packing more efficient if it holds.

3.3 The Framework

A **Fascicle Processor** is a single physical ARM core on a SpiNNaker chip. It may be host to zero, one or more fascicles in the simulation process. The intention is that of the *n* ARM cores on a chip, *n-1* will be Fascicle Processors. (The other ARM will be used for housekeeping functions.)

The overall APR structure is fairly conservative, and outlined in figure 4. It is a heuristic; the size of the datasets makes iteration a very expensive operation, and so the design intention is that a neural circuit will pass through the design flow only once. Feedback of any kind is to be avoided, but if this is not possible, the feedback loops have been arranged in order of computational expense:

fb1: Should never be necessary anyway.
fb2: Is cheap and may never even be necessary - see later.
fb3: Is cheap, but the usefulness is dubious.
fb4: The Loop of Last Resort: expensive, but allows the system to expand onto unused SpiNNaker chip sites if any exist.

3.3.1 Neurone to Fascicle Mapping

Here we take the input neurone circuit (figure 2, for example), and partition it between the Fascicle Processors. In outline, the algorithm - based on the Kernighan-Lin partitioning scheme 7 - is as follows:

1. The input graph G is bisected randomly (in terms of synapse count) into two subgraphs, A and B. (These are potential fascicles - we note that, in general, each will be far too big to fit into a single SDRAM, certainly for the first few recursion levels of the algorithm).
2. Define some size limit, h, corresponding to the approximate capacity of a Fascicle Processor.
3. Find the neurone (in A or B) that (a) is unflagged, and (b) that would make the biggest improvement (which may, actually, be the smallest degradation) to the penalty function d() if it were to be moved to the opposite subgraph.
   If Δd() is an improvement, and does not violate the fascicle size limit h, **then** {move it and flag it, return to the start of step 3}
   // Only here if the best Δd() is actually a degradation
   If it was the first attempt (i.e. no neurones are flagged) **then** stop, **else** clear flags and return to the start of step 3.
4. Recursively apply step 3, replacing G with A and B at each level. The size of G will (approximately) halve at each recursion level; a recursive branch can terminate when h < SDRAM size (i.e. we have created a fascicle that will fit into a node SDRAM).

The nature of the penalty function, d(), is worthy of some comment. In the traditional (electronic) context, it will represent the number of interconnects that cross the subgraph partition (i.e. pass between A and B). Here, however, because of the way the data is packed into the SDRAM, we are not attempting to *minimise* the cut interconnects, we are attempting to *even out* the density of fascicle-fascicle interconnect. In essence, we look at each neurone in turn (subject to the restrictions in step 3 above), and see what effect would be had on the cut-line count if it were to be moved to the opposing subgraph. The neurone that is chosen is the one that minimises the total standard deviation of the cut-line counts.

3.3.2 SDRAM Data

The output from the algorithm of the previous section is a set of bitmaps, which represent the connectivity of the neural circuitry, plus the state. These are loaded into the SDRAM of figure 1.

3.3.3 Fascicle to Fascicle Processor Mapping

Having partitioned the neural network into fascicles, it is now necessary to map these onto the individual fascicle processors, as in figure 2. The partitioning achieved by the last algorithm simply divided the neural aggregate up into fascicles, but the attributes of geometric position were not assigned to the neurones. This is done using a combination of Lees algorithm and force-directed placement.

The force-directed algorithm is ideal for this; it has approximately linear complexity and delivers an acceptable placement extremely quickly

3.3.4 Routing Geometry

The derivation of the individual packet trajectories over the system is simple. For a given point-to-point route, there will be a maximum of six 'shortest' routes (recall that the toroidal layout has no edges) and an arbitrary number of longer possibilities. The only real selection criterion here is the capacity of the routing tables at the inflection nodes. Each route is examined in turn, and that with the *lowest line integral of routing table occupancy* chosen 8. Thus the interconnect route density is kept as even as possible over the routing surface.

3.3.5 Routing Tables

The final step in the process of loading the SpiNNaker ensemble is the generation of the data to be contained in the routing tables. A 32-bit source key is input to a 1024 x 32 bit tristate CAM. Hits are written to a 1024 x 1 bit hit register. All but the most significant single bit in this register are discarded, and this single remaining bit treated as a 1024 bit 1-hot and passed into an address encoder. This generates a 10 bit binary equivalent, which drives a 1024 x 26 bit lookup RAM. The 26 bit word so generated consists of a 6-bit nibble and a 20-bit nibble. The 6-bit nibble represents an n-hot external link indicator (0-5) to which the packet is forwarded (for example 010110 would cause the packet to be routed to external links 1, 2 and 4). The 20-bit nibble represents an n-hot internal Fascicle Processor address (0-19) to which the packet will be forwarded, triggering an interrupt as it arrives. (For example, 00001000100100000000 will cause packets to go to Fascicle Processors 8, 11 and 15 *on the current chip*. It is easy to see how packets may be duplicated by this mechanism. The 1024 x 32 bit tristate CAM is implemented as a 1024 x 32 bit binary CAM and a 1024 x 32 bit binary RAM. The RAM simply holds a bit mask indicating the position of the "don't cares" in the CAM. The corresponding bits in the CAM will *actually* be '0' or '1', but will never be read.

4 Final Remarks

The design, development and realisation of a customised APR tool suite for the SpiNNaker system is a significant undertaking, requiring resources comparable to that of the hardware design. Like the hardware, the system is not yet complete, although the hope and intention is that the alpha release will coincide with the delivery of the prototype silicon.

The detailed design and development of the SpiNNaker hardware is an extremely complicated piece of electronic design, and this naturally makes heavy use of EDA tool suites. A simulation model of the chip has been built, but unfortunately (perhaps not unsurprisingly) this is unable to cope with the simulation of neural systems of any realistic size or complexity. It should be noted, however, that the datastructures required to support a bespoke behavioural simulator are all present in the tools described here; the addition of a behavioural simulation capability is not seen as a vast undertaking.

Acknowledgements

The SpiNNaker project is supported by the Engineering and Physical Sciences Research Council, UK, ARM and Silistix.

References

1. Furber, S.B., Temple, S., Brown, A.D.: On-chip and inter-chip networks for modelling large-scale neural systems. In: Proc. International Symposium on Circuits and Systems, ISCAS-2006, Kos, Greece (May 2006)
2. Furber, S.B., Temple, S.: Neural systems engineering. J. R. Soc. Interface 4(13), 193–206 (2007)
3. Plana, L.A., Furber, S.B., Temple, S., Khan, M.M., Shi, Y., Wu, J., Yang, S.: A GALS infrastructure for a massively parallel multiprocessor. IEEE Design and Test of Computers 24(5), 454–463 (2007)
4. Quinn, N.R.: The placement problem as viewed from the physics of classical mechanics. IEEE Circuits and Systems CAS-26(6), 173–178 (1979)
5. Vecchi, M.P., Kirkpatrick, S.: Global wiring by simulated annealing. IEEE Transactions on Computer-Aided Design TCAD-2(4), 215–222 (1983)
6. Lee, C.V.: An algorithm for path connections and its applications. IRE Transactions on Electronic Computers, 346–364 (1969)
7. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Sys. Tech. Journal 49, 291–308 (1970)
8. Brown, A.D.: Automated placement and routing. Computer-aided design 20

The Deferred Event Model for Hardware-Oriented Spiking Neural Networks

Alexander Rast*, Xin Jin, Mukaram Khan, and Steve Furber

School of Computer Science, University of Manchester
Manchester, UK M13 9PL
{rasta,khanm,jinxa}@cs.man.ac.uk,
steve.furber@manchester.ac.uk
http://www.cs.manchester.ac.uk/apt

Abstract. Real-time modelling of large neural systems places critical demands on the processing system's dynamic model. With spiking neural networks it is convenient to abstract each spike to a point event. In addition to the representational simplification, the event model confers the ability to defer state updates, if the model does not propagate the effects of the current event instantaneously. Using the SpiNNaker dedicated neural chip multiprocessor as an example system, we develop models for neural dynamics and synaptic learning that delay actual updates until the next input event while performing processing in background between events, using the difference between "electronic time" and "neural time" to achieve real-time performance. The model relaxes both local memory and update scheduling requirements to levels realistic for the hardware. The delayed-event model represents a useful way to recast the real-time updating problem into a question of time to the *next* event.

1 Real-Time Neural Networks: The Update Timing Challenge

Accurately modelling the continuous-time behaviour of large neural networks in real time presents a difficult computational choice: when to update the state? Conventional sequential digital processing usually prohibits real-time updates except on the largest, fastest computers, but dedicated parallel neural network hardware needs some time model. Broadly, two different architectures have become popular. One, the neuromorphic approach, e.g. [1], circumvents the state update problem altogether by using continuous-time analogue circuitry - at the price of hardwiring the model of computation into the system. The other, the parallel neurocomputer approach, e.g. [2] retains general-purpose digital processing but attempts to use a neural-specific, multiprocessor architecture - with some loss of speed and accuracy of state update. An ideal approach would combine the process abstraction capabilities of digital devices with the asynchronous-time model of analogue devices. SpiNNaker, a chip using event-driven asynchronous

* Corresponding author.

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 1057–1064, 2009.

digital circuitry timed by its input rather than by an instruction clock, makes this feasible for spiking neural networks having real axonal and synaptic delays, by abstracting a spike to an event. During the time between events, it is in effect "outside time", and can perform necessary background processing without affecting the model update. Having previously demonstrated a basic neural implementation on SpiNNaker, here we present a general method to maintain accurate temporal dynamics by deferring pending updates until the next input event. The method realises a solution to the update scheduling problem that suggests the possibility of practical large-scale, real-time neural simulation.

2 Deferred Update Dynamics

Real dynamic properties of neural networks allow us to relax the timing requirements dramatically. Most models, whether using temporal [3] or rate coding [4], [5], assume that the spike timing irrespective of shape determines the information coding. A typical active neuron fires at ∼10-20 Hz up to a maximum of ∼100 Hz [6]. Only a small number of a given population of neurons, typically ∼1-0.1%, will be active at any time, with 10% a reasonable upper limit. For a "typical" neuron containing 5000 dendritic connections with 1% activity, spiking at 10 Hz, we therefore expect an average input rate of 500 events (input spikes) per second, requiring an update rate of only 2ms. A worst-case situation: 100k inputs, 10% activity, 100 Hz firing rate, would require 1μs update rate. These are leisurely rates for typical digital processors running at hundreds of MHz. With real neurons having axonal delays, usually of the order of 1-20 ms, if the processor can propagate the required updates following an event in less time than the interval between events that affect a given output, it can use that time difference to defer the event processing until the occurrence of the *next* event. Simple time-domain multiplexing [7] is an established approach, but in addition, the processor can make use of the "dead space" to wait on contingent input events that may likewise affect the output. By performing temporal reordering of input events, it can ignore the fine-grained order of inputs. In Fig. 1, axons have finite delays. The processor schedules the updates, and the neuron can respond to further potential future input events with nonzero axonal delays, reordering them as necessary so that the neuron exhibits the proper dynamic behaviour even with nondeterministic input order. Delayed event processing thus allows more than time multiplexing; it decouples the temporal dynamics of the processing system from the temporal dynamics of the model.

3 Implementation of a Neuron and Synapse on SpiNNaker

3.1 The SpiNNaker Hardware System

We have developed a universal spiking neural network chip multiprocessor, SpiNNaker (fig. 2), to support large scale real-time simulation [8]. 20 ARM968 processor cores with dedicated 64KB local memory implement neurons and their associated dendrites. A large off-chip SDRAM memory device stores synaptic data,

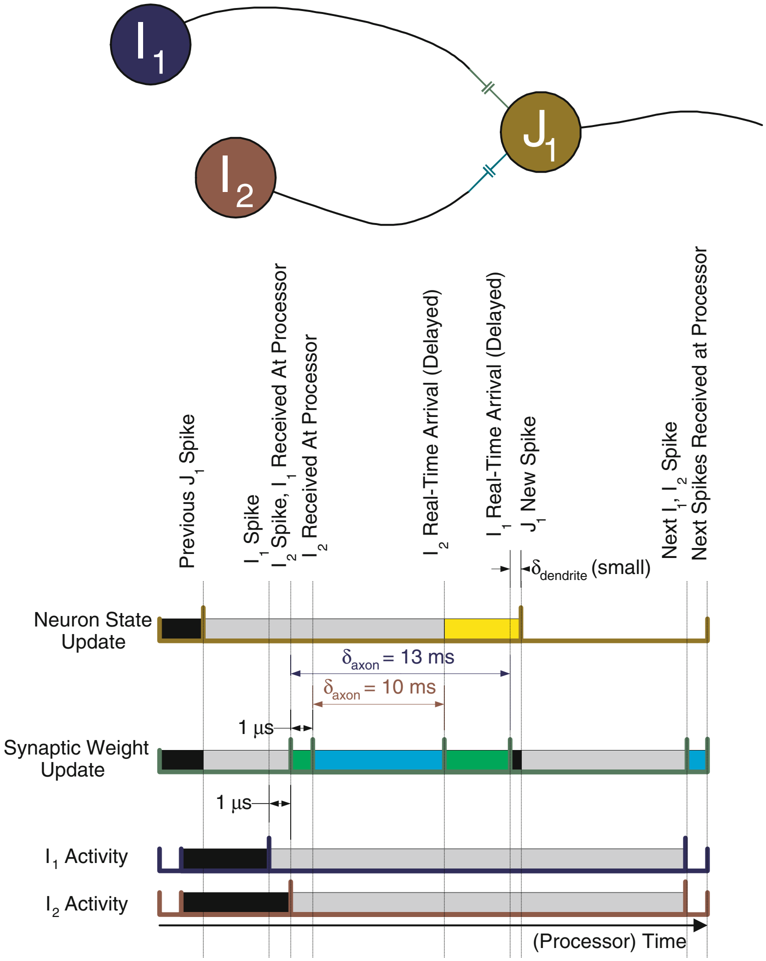


Fig. 1. Deferred Event Processing. At top is a very simple network, with schematic event spike trains at the bottom. The spike trains further show time before the current event (dark regions), time to process the current event (medium regions), and time when the processor can defer the processing (light regions). Neuron J_2 receives inputs from two other neurons I_1 and I_2 with delays of 13 ms and 10 ms respectively. Suppose that the processor performing the modelling can propagate the output in 1 μs. Neuron I_1 fires first in the model, with I_2 following close behind it after a 1 μs delay. Thus the corresponding events arrive at J_2 after 1 μs and 2 μs respectively. But because of the real delays in the model, the processor will not actually need the input from neuron I_1 for another 12.999 ms. Thus at 1 μs, it need do nothing with that input event other than record it. In particular, it can wait for input I_2, and when that event occurs at 2 μs, it can schedule its update event 9.999 ms into its future, with the update event for input I_1 still 12.998 ms in the future.

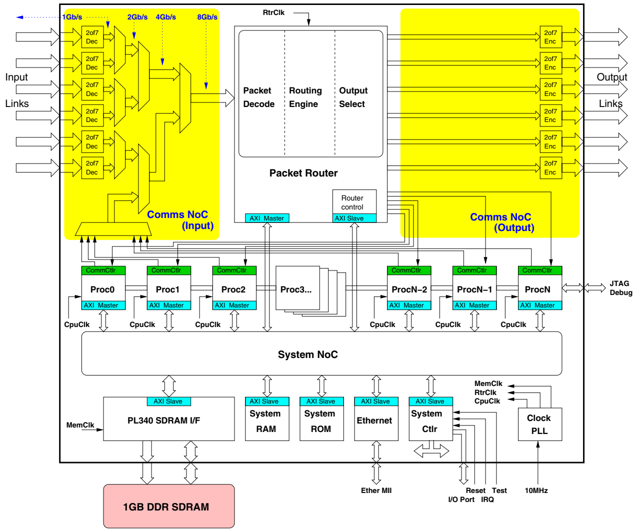


Fig. 2. SpiNNaker design, showing router, processors, and internal NoC components

made to appear "virtually local" to the processor using an optimised DMA controller in concert with a 1Gb/s internal asynchronous Network on Chip (NoC) [9]. A configurable 6Gb/s per node global asynchronous packet-switching network, using an on-board source-based associative router together with an event-driven communications controller, implements axons and the associated connectivity map [10]. Using the "address-event representation" (AER) format [11], SpiNNaker signalling abstracts the neural action potential into a single discrete point event happening in zero time, a "spike". An AER packet simply records the source of the spike (and optionally an extra "payload" data field), to route it to target neurons potentially distributed over a system of up to 64K chips. Both the physical and temporal details of SpiNNaker hardware are invisible to the neural network model, allowing the device to be loaded and configured to support virtually any model of dynamics with any topology [12].

3.2 Neuron

As a SpiNNaker case study, we have implemented the Izhikevich model [13] neuron because it is simple, instruction-efficient, and exhibits most of the dynamic properties of biological neurons. Since the ARM968 processor cores contain no floating-point support, and studies [14], [15], [16] indicate that 16-bit precision is adequate for most purposes, we represent all variables and parameters with 16-bit fixed-point variables. A 16-element array (fig. 3) represents input stimuli. Each element, representing a time bin having model axonal delay from 1 ms to 16 ms, carries a stimulus. When a spike packet with a delay of δ_m ms arrives at t_n ms, the processor increments the value in the $t_n+\delta_m$ ms time bin by the connection weight of the input synapse, deferring the neural state update until

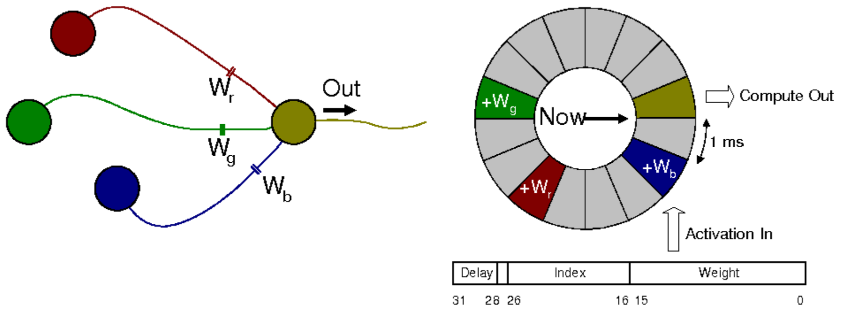


Fig. 3. SpiNNaker Neuron Binned Input Array. Inputs arrive into the bin corresponding to their respective delay. Output is computed from the bin pointed to by "Now".

the global simulation time reaches $t_n + \delta_m$ - the time when the input "actually" arrives. During state update, we extract the accumulated stimulus for the current time bin from the data structure of one neuron and pass it along with the (static) parameters to the two Izhikevich equations, storing the new neural state values back to the data structure. For each update, we test the value of the membrane potential. If it reaches its reset value, the neuron fires and the processor will issue a multicast packet including the processor ID and the neuron ID. The processor maintains real-time accuracy as long as its neurons can be updated before shifting to the next bin position - nominally 1 ms intervals in our model.

3.3 Synapse

For the synapse, we have used the exponential spike-timing dependent plasticity (STDP) model [3]. Weight updates exploit the fact that the updated value will not be required until the input event *after* the current one. Each source neuron has an associated "row" in SDRAM containing a time stamp and target-indexed weight entries only for those target neurons on the local processor with nonzero weights (fig. 4), permitting a single table lookup per input event to retrieve the synapse data. For the time stamp we have used a 64 ms two-phase time window comparable to the $\sim$ 50 ms experimentally observed STDP time window [17], permitting 4 ms spike-time resolution within a 32K × 64ms coarse time period, enough to capture about 35 minutes without time rollover. With the arrival of an input, we update the time stamp and compute the *previous* weight update as follows: Retrieve the synapse row. For each connected neuron, compare its local coarse time stamp (postsynaptic spiking times) against the retrieved time stamp (presynaptic activation times). If the coarse time stamp matches, perform the weight update rule the model specifies, using the fine time stamp. (By coding the weight update as the power-of-2 bit-position in the fine time stamp, the processor can compute the update with a series of simple shift-and-add operations). If the neuron fires, update its local time stamp. If the delayed input time is within a synaptic window of the current time stamp, place a 1 in the stored (presynaptic) time stamp at the bit position corresponding to the time offset. Write back the

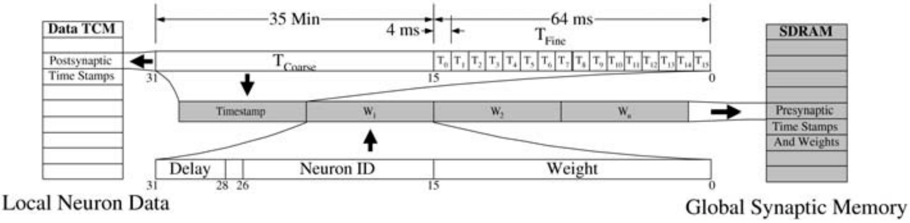


Fig. 4. Synapse Data Format. Each of the data words is 32 bits wide. Synapse rows in global memory have multiple weights and a time stamp word. Weights contain a 16-bit weight field and a 16-bit parameter field. In our experiments we used 4 bits of the parameter field to represent synaptic delays and 11 to index the target neuron. Each row in SDRAM has one time stamp (time of last presynaptic event), and each neuron in local memory a second (time of last postsynaptic spike). The time stamp has 2 fields, a fine and coarse time stamp. The fine time stamp is bit-mapped into 16 equal time segments with resolution $\frac{I_w}{16}$, where I_w is the total length of the time window, containing a 1 if there was an input activation at that real-time point within the interval. The coarse time stamp is an integer with a resolution of $\frac{I_w}{\phi}$, where ϕ is a (small) power-of-2 constant representing a time phase within the window I_w. The model updates this time stamp if a transmission event t_{new} occurs at $t_{new} - t_{last} \geq \frac{I_w}{\phi}$, where t_{last} is the current value of the coarse time step.

updated synapses and time stamp to memory. The essential timing requirement is that the update computation takes less time than the time between inputs on the same synapse.

4 Application to System Modelling

We have created a complete system-level simulation modelling multiple SpiNNaker chips using SystemC, and a cycle and ARM Instruction Set accurate simulation using ARM SoC designer, to verify the design, support early application development, and test the overall chip/system behaviour. We calibrated cycle-accurate SystemC behaviour of modelled chip components against their Verilog-based simulations used for chip fabrication. We performed case studies to test the functionality of the chip, the routing fabric, and the system under various scenarios. To verify the delayed-event model we implemented the Izhikevich [13] neural dynamic model using Xin Jin, et al.'s scheme [14] and successfully tested a small population of neurons (fig. 5). We tested the physical delays of the neurons in the population to determine the time margin a typical input would have to complete its processing (fig. 5). According to our simulations the update time for one neuron is about 240 ns. With 1000 neurons, this corresponds to a total update time of 0.24 ms - comfortably within the worst-case 1 ms delay margin even with nondeterministic arrival times. By delaying the state update, we effectively compress these computations into the time immediately after an update event, releasing the slack for further processes such as synaptic updating.

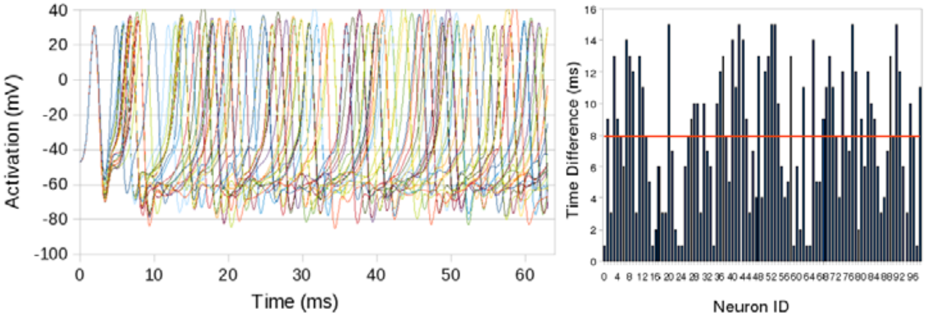


Fig. 5. Neuron spike traces (L) and delay margins (R). Traces are the output response to delayed inputs. The bars show the time difference between the modelled axonal delay and the hardware delay, measured from spike packet transmission time to packet reception time. Worst-case delay margin is 1 ms. Mean delay margin is 7.93 ms.

5 Conclusions

We have developed a model capable of achieving biologically realistic real-time performance on event-driven neural network hardware, with nondeterministic signal timing. The deferred-update model permits not only real-time signal timing resolution but also more efficient processor utilisation, since each processor can schedule its updates according to the model-time rather than the hardware-time event sequence. SpiNNaker has three dimensions of configurability, letting the user specify the topology, processing dynamics, and temporal dynamics according to the needs of the model, rather than those of the hardware. Such a "neuromimetic" system mitigates both against instant hardware obsolescence and the need to make hard decisions about what neural models to experiment with in view of the hardware available. It is, perhaps, therefore a more viable architecture for future neural systems than approaches that impose hard physical constraints. Our hope is that SpiNNaker might encourage dialogue between the biological scientists and the computer scientists, to link phenomenological observations to behavioural models of computation. Such systems could promote a two-way flow of learning, discovering on the one hand more about how the brain functions and on the other new and alternative computing models.

Acknowledgements. We would like to thank the Engineering and Physical Sciences Research Council (EPSRC), Silistix, and ARM for support of this research. S.B. Furber is the recipient of a Royal Society Wolfson Merit Award.

References

1. Indiveri, G., Chicca, E., Douglas, R.: A VLSI Array of Low-Power Spiking Neurons and Bistable Synapses With Spike-Timing Dependent Plasticity. IEEE Trans. Neural Networks 17(1), 211–221 (2006)

2. Mehrtash, N., Jung, D., Hellmich, H., Schönauer, T., Lu, V.T., Klar, H.: Synaptic Plasticity in Spiking Neural Networks (SP^{2}INN): a System Approach. IEEE Trans. Neural Networks 14(5), 980–992 (2003)
3. Gerstner, W., Kempter, R., van Hemmen, J.L., Wagner, H.: A Neuronal Learning Rule for Sub-millisecond Temporal Coding. Nature 383(6595), 76–78 (1996)
4. Shadlen, M.N., Newsome, W.T.: The Variable Discharge of Cortical Neurons: Implications for Connectivity, Computation, and Information Coding. J. Neuroscience 18(10), 3870–3896 (1998)
5. Masuda, N., Aihara, K.: Duality of Rate Coding and Temporal Coding in Multilayered Feedfoward Networks. Neural Computation 15(1), 103–125 (2003)
6. Dayan, P., Abbott, L.: Theoretical Neuroscience. MIT Press, Cambridge (2001)
7. Goldberg, D., Cauwenberghs, G., Andreou, A.: Analog VLSI Spiking Neural Network With Address Domain Probabilistic Synapses. In: Proc. 2001 Int'l Symp. Circuits and Systems (ISCAS 2001), pp. 241–244 (2001)
8. Furber, S.B., Temple, S.: Neural Systems Engineering. J. Roy. Soc. Interface 4(13), 193–206 (2007)
9. Rast, A., Yang, S., Khan, M.M., Furber, S.: Virtual Synaptic Interconnect Using an Asynchronous Network-on-Chip. In: Proc. 2008 Int'l Joint Conf. Neural Networks (IJCNN 2008), pp. 2727–2734 (2008)
10. Plana, L.A., Furber, S.B., Temple, S., Khan, M.M., Shi, Y., Wu, J., Yang, S.: A GALS Infrastructure for a Massively Parallel Multiprocessor. IEEE Design & Test of Computers 24(5), 454–463 (2007)
11. Lazzaro, J., Wawrzynek, J., Mahowald, M., Silviotti, M., Gillespie, D.: Silicon Auditory Processors as Computer Peripherals. IEEE Trans. Neural Networks 4(3), 523–528 (1993)
12. Khan, M.M., Lester, D., Plana, L., Rast, A., Jin, X., Painkras, E., Furber, S.: SpiNNaker: Mapping Neural Networks Onto a Massively-Parallel Chip Multiprocessor. In: Proc. 2008 Int'l Joint Conf. Neural Networks (IJCNN 2008), pp. 2849–2856 (2008)
13. Izhikevich, E.: Simple Model of Spiking Neurons. IEEE Trans. Neural Networks 14, 1569–1572 (2003)
14. Jin, X., Furber, S., Woods, J.: Efficient Modelling of Spiking Neural Networks on a Scalable Chip Multiprocessor. In: Proc. 2008 Int'l Joint Conf. Neural Networks (IJCNN 2008), pp. 2812–2819 (2008)
15. Wang, H.P., Chicca, E., Indiveri, G., Sejnowski, T.J.: Reliable Computation in Noisy Backgrounds Using Real-Time Neuromorphic Hardware. In: Proc. 2007 IEEE Biomedical Circuits and Systems Conf. (BIOCAS 2007), pp. 71–74 (2007)
16. Daud, T., Duong, T., Tran, M., Langenbacher, H., Thakoor, A.: High Resolution Synaptic Weights and Hardware-in-the-Loop Learning. In: Proc. SPIE - Int'l Soc. Optical Engineering, vol. 2424, pp. 489–500 (1995)
17. Markram, H., Tsodyks, P.: Redistribution of Synaptic Efficacy Between Neocortical Pyramidal Neurons. Nature 382(6594), 807–810 (1996)

Particle Swarm Optimization with SIMD-Oriented Fast Mersenne Twister on the Cell Broadband Engine

Jun Igarashi[1], Satoshi Sonoh[1], and Takanori Koga[2]

[1] Graduate School of Life Science and Systems Engineering
Kyushu Institute of Technology
2-4 Hibikino, Wakamatsu-ku, Kitakyushu, 808-0196, Japan
{igarashi@,sonoh@edu.}brain.kyutech.ac.jp
[2] Graduate School of Science and Engineering
Yamaguchi University
1677-1 Yoshida, Yamaguchi-shi Yamaguchi, 753-8511, Japan
koga@ic.sci.yamaguchi-u.ac.jp

Abstract. We introduce a processing performance of Particle Swarm Optimization with SIDM-oriented Fast Mersenne Twister on the Cell Broadband Engine. Extreme-high processing performance is demanded for solving very complex optimization problem in a small amount of time. In this research, we verified the effectiveness of employing SIMD-oriented Fast Mersenne Twister on the Cell Broadband Engine for the processing of Particle Swarm Optimization by numerical simulations.

1 Introduction

Optimization problem is the problem of finding out the best solution from all feasible solutions. Particle Swarm Optimization (PSO) [1], which is a meta-heuristic inspired by a swarm of insects or a pod of fish, shows excellent performance for optimization problems and has been successfully applied to function optimization [2] and neural network training [3]. PSO achieves to search the best solution by using several particles that represent candidate solutions. PSO has a good feature that the algorithm is described by simple equations that update positions of the particles using a pseudo random number. However, its computational cost drastically increases according to the number of the particles, search dimensions and search trials. PSO requires vast amount of calculations including a generation of pseudo random number. The generation of pseudo random number is very important to the performance of PSO and relatively occupies a large amount of processing time in PSO processing. Thus, a fast generation method of the pseudo random number is essential for improvement of PSO in aspect of the processing time.

SIMD-oriented Fast Mersenne Twister (SFMT) is designed for fast generation of the pseudo random number using Single Instruction Multiple Data (SIMD) operation and multi-stage pipelines of modern CPUs. SFMT generates pseudo random numbers extremely fast compared with conventional ones. Furthermore, quality of the generated numbers is superior as random numbers. In order to bring out the performance of PSO with SFMT, a suitable processing unit should be used. In this paper, we propose

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 1065–1071, 2009.

to apply the Cell Broadband Engine (Cell/B.E.) processor to PSO with SFMT. The Cell/B.E. processor is a heterogeneous processor that targets video games, multimedia applications, and scientific computation. The Cell/B.E. has excellent floating-point processing performance compared with other old supercomputers.

Additionally, the Cell/B.E. is available for a scientific research with a reasonable price because it is mounted on the game console, PLAYSTAION3. In this paper, we implement PSO algorithm with SFMT on the Cell/B.E. processor for speed-up of PSO. Finally, we show that the Cell/B.E. effectively perform PSO with SFMT compared with a general processor.

2 Implementation of Particle Swarm Optimization with SIMD-Oriented Fast Mersenne Twister on the Cell Broadband Engine

2.1 Particle Swarm Optimization and DeJong's Benchmark

Fig. 1 shows the schematic picture of PSO. Each individual in the particle swarm is composed of three *d*-dimensional vectors, where *d* is the dimensionality of the search space. The current position, the previous best position and the velocity are denoted by $\boldsymbol{x}_i$, $\boldsymbol{p}_i$ and $\mathbf{v}_i$, respectively. Here, *i* is index of the individual. In the particle swarm optimization process, the velocity of each particle is iteratively adjusted so that the particle stochastically oscillates around $\boldsymbol{p}_i$ and $\boldsymbol{p}_g$. $\boldsymbol{p}_g$ is the best of all $\boldsymbol{p}_i$. At each step, the desired optimization fitness function in *d* variables is evaluated, then, $\boldsymbol{p}_i$ and $\boldsymbol{p}_g$ are renewed. The equation of PSO is described as follow,

$$\begin{cases} \mathbf{v}_i^{k+1} = \gamma \mathbf{v}_i^k + c_1 \mathbf{U}(0,\phi_1) \otimes \left(\mathbf{p}_i^k - \mathbf{x}_i^k\right) + c_2 \mathbf{U}(0,\phi_2) \otimes \left(\mathbf{p}_g^k - \mathbf{x}_i^k\right) \\ \mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^k . \end{cases}$$

Here, gamma = 0.99, C1 = 0.9 and C2 = 0.9 are parameters which govern search policy of each particle. *k* is time step. U() is a vector of pseudo random numbers uniformly distributed in [0, φ]. ⊗ is component-wise multiplication.

As a benchmark of PSO, DeJong's benchmarks [4] were used. In the experiments, 100 and 500 iterations were performed for F1-F3 and F4-F5, respectively. 2400 trials were done in each benchmark.

2.2 SIMD-Oriented Fast Mersenne Twister

SFMT is proposed by M. Matsumoto and M. Saito in 2006 [5]. SFMT is a linear feed back shift register type pseudo random number generator. SFMT is designed considering new features of modern CPUs, such as Single Instruction Multiple Data (SIMD) operation and multi-stage pipeline processing.

We coded SFMT for the Cell/B.E. by modifying SFMT version 1.3.3 for PowerPC Altivec provided on the website [6]. In the case of using SFMT on the Core 2, SIMD processing was disabled. The fill_array32 function was used in SFMT for batch processing of pseudo random number generation. The rand() function of standard library of ANSI C was used for performance comparison with SFMT.

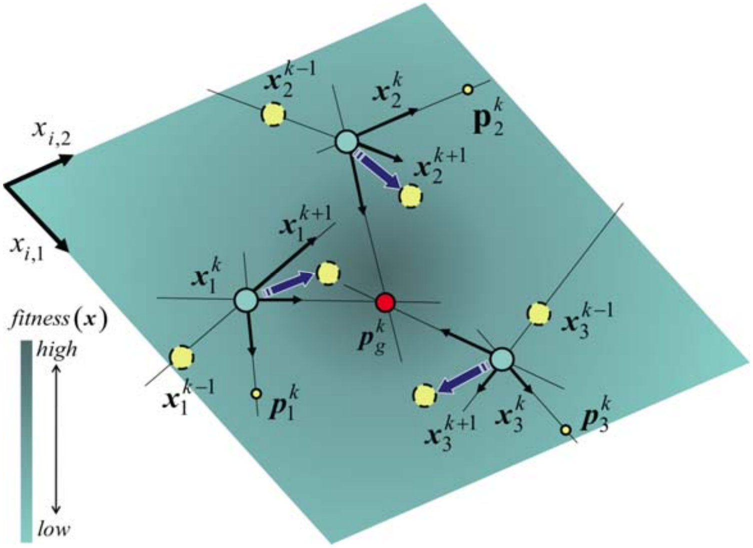


Fig. 1. The schematic picture of PSO. Particles represented as circles that search the optimum value in cooperation with other particles.

2.3 Cell Broadband Engine Processor

The Cell/B.E. is a heterogeneous microprocessor that results from a joint effort among the Sony Group, Toshiba and IBM. The Cell/B.E. has a PowerPC processor Element (PPE) designed for general-purpose processing and 8 Synergistic Processor Elements (SPEs) designed for high performance floating-point computation [7]. In the Linux programming environment of PLAYSTATION3, the one PPE and the six SPEs of the Cell/B.E. are available. The peak performance of the Cell/B.E. processor of PLAYSTATION3 is about 200 GFLOPS. In this research, we mainly used the SPEs for PSO processing except for starting and ending process of the SPE threads by the PPE. Numerical calculation was performed with single precision. Table.1 shows equipment of the Cell/B.E. Machine and Core 2 Machine.

Table 1. The equipment of the Cell/B.E. Machine and the Core 2 Machine

	Cell/B.E. Machine: PLAYSTATION3	PC: IBM PC/AT Compatible
CPU	Cell B.E. 3.2 GHz (PPE +6SPE)	Core 2 Duo 2.66GHz (1core usage)
Memory	XDR 256MB	DDR2 2GB
OS	FedoraCore6 (kernel 2.6.18-53.1.4.el5)	CentOS5 (kernel2.6.18-53.1.4.el5)
Compiler	GCC 4.1.1 for Cell B.E (-O3)	Intel Compiler ICC (-O3)
Library	Cell SDK 2.1, libspe2, SIMD Math Library	

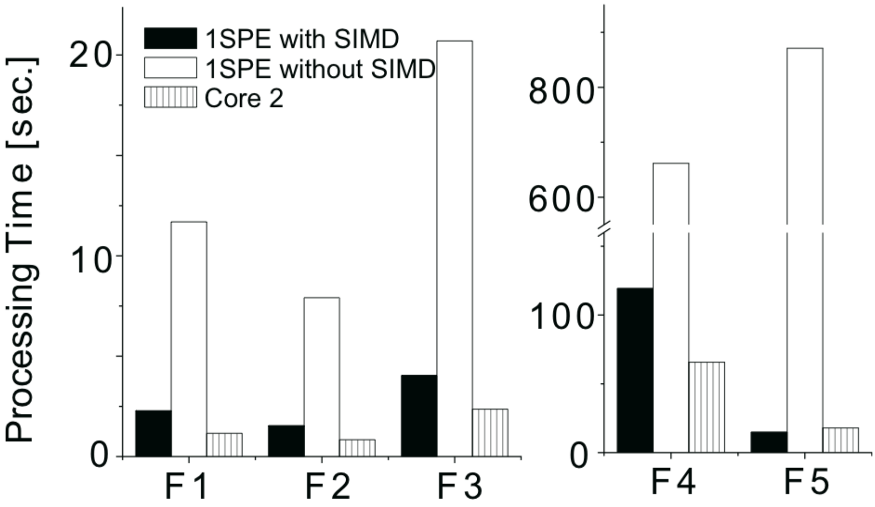


Fig. 2. The reduction of processing time by using SIMD processing on the one SPE in DeJong's benchmark

3 Experimental Results and Discussions

First, we compared the processing time using the one SPE of the Cell/B.E. with that using the Core 2 processor when the rand() function of ANSI C was used for pseudo random number generation in PSO processing. Fig. 2 shows the results of DeJong's benchmarks of PSO using the one SPE with SIMD instructions, the one SPE without SIMD instructions and the Core 2 with normal C program. In the SIMD processing, 4

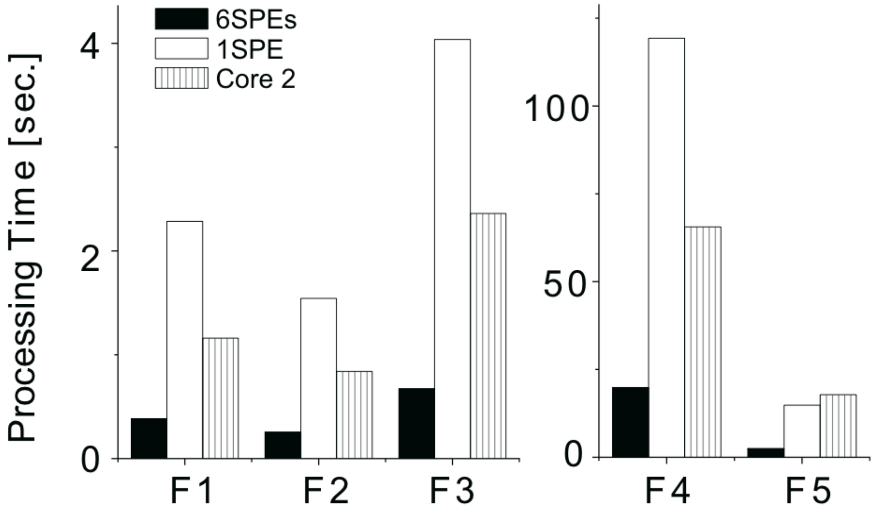


Fig. 3. The performances calling with increase in the number of SPE

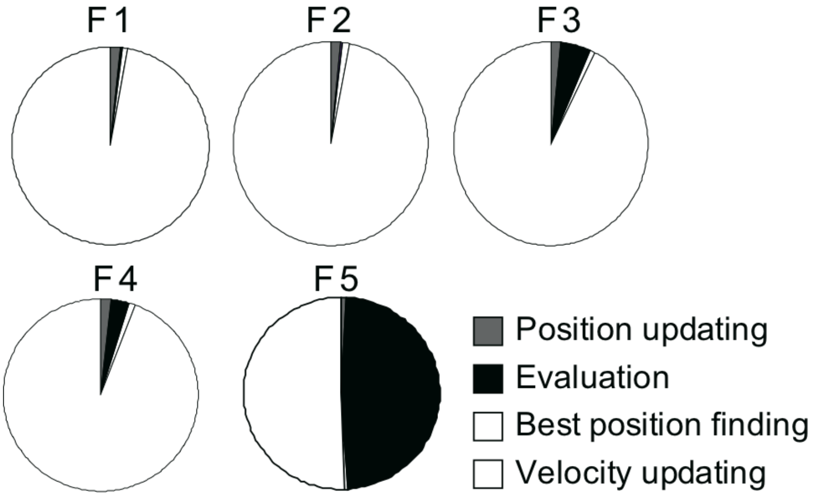


Fig. 4. Pie charts for processing time ratios

trials were assigned to each element of one vector, and performed in parallel. In the all benchmarks, by using SIMD, the processing times of Cell/B.E. were shortened more than 5 times. These were because of parallelism effect by SIMD instruction and efficient execution of intrinsic functions for the SIMD instruction. The processing times using the one SPE with SIMD instructions took twice as much as that of the Core2.

Fig. 3 shows the results of DeJong's benchmarks of PSO when the one SPE, the six SPEs or the Core 2 was used with the rand() function. The six SPEs computed PSO 6 times faster than the one SPE. This performance scaling occurred due to parallel execution of independent trials on the six SPEs. As compared with the Core 2, the six SPEs took 1/7 to 1/3 of processing time. Fig. 4 shows a pie chart showing processing time ratio. PSO processing mainly consisted of 4 parts, updating of particle position, evaluation, finding of the best value and updating of velocity. The longest time in all benchmarks was taken in the part of updating of velocity that mainly performed pseudo random number generation. Speeding up of random number generation should be effective for shortening processing time of PSO.

Table 2. The processing time ratios between the Cell/B.E. and the Core 2 with SFMT or the rand() function

	F1	F2	F3	F4	F5
Core 2-rand/ Cell-SFMT	41.4	42.0	29.1	33.5	13.9
Core 2-SFMT/ Cell-SFMT	13.9	15.0	13.7	12.1	12.0
Core 2-rand/ Cell-rand	3.0	3.3	3.5	3.3	7.2

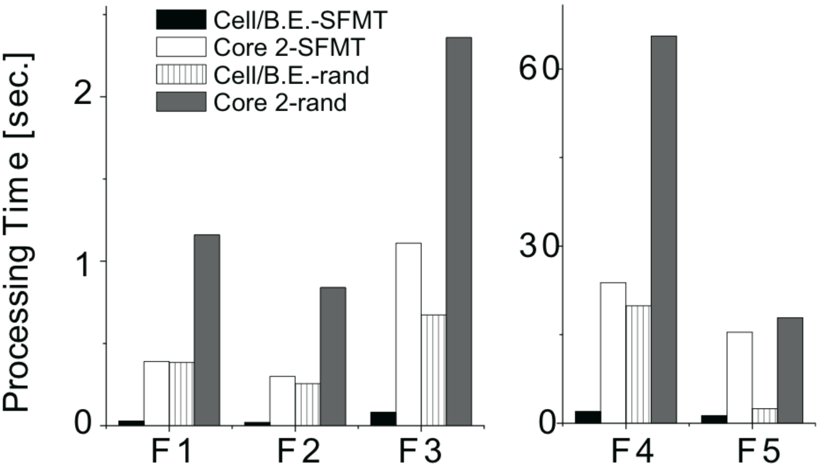


Fig. 5. The high speed processing of PSO by the Cell/B.E. with SFMT

Next, we introduced SFMT into PSO processing. Fig. 5 shows processing times of DeJong benchmarks of PSO using the six SPE of the Cell/B.E. or the Core 2. In the cases, SFMT or rand() function was used for pseudo random number generator. In the both case of the Cell/B.E. and the Core 2, processing times were drastically shorten by employing SFMT as compared with the rand() function. Combination of SFMT and the Cell/B.E. performed fastest computation in the all benchmarks. Table 2 shows processing time ratios between the Cell/B.E. and the Core 2 shown in Fig. 5. The combination of the Cell/B.E. and SFMT were approximately 14 to 42 times faster than the combination of the Core 2 and the rand() function (see Table 2 row 1). Table 2 also shows effectiveness of combination of SFMT and the Cell/B.E. When the rand() function was used for both processors, the Cell/B.E. calculated faster just 3 to 7 times than the Core 2 (see Table 2 row 3). On the other hand, when SFMT was used for the both processors, the Cell B.E. calculated 12 to 15 times faster than the Core 2 (see Table 2 row2). This result suggests that SFMT brought out high floating-point performance of the Cell/B.E. because of its optimization for SIMD and pipeline feature, comparing with the rand() function that performed scalar instruction and was not specially designed for pipeline features.

4 Conclusion

In this paper, we showed that fast processing of PSO using SFMT on the Cell/B.E.. By only applying multi-core processing and SIMD processing technique of Cell/B.E. for PSO processing, PSO processing by the Cell/B.E. were from 3 to 7 times faster than the general processor, Core 2. Furthermore, by applying SFMT on the Cell/B.E., the Cell/B.E. calculated PSO processing faster from approximately 14 to 42 times than Core 2 using rand() function.

Acknowledgments. This work is partially supported by Kitakyushu Foundation for the Advancement of Industry, Science and Technology.

References

1. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: IEEE the International Conference on Neural Networks, pp. 1942–1948 (1995)
2. Clerc, M.: Particle Swarm Optimization. ISTE USA, USA (2006)
3. Cai, X., Zhang, N., Venayagamoorthy, G.K., Wunsch, D.C.: Time series prediction with recurrent neural networks trained by a hybrid PSO–EA algorithm. Neurocomputing 70, 2342–2353 (2007)
4. DeJong, K.A.: An analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan (1975)
5. Saito, M., Matsumoto, M.: SIMD-oriented Fast Mersenne Twister: a128-bit Pseudorandom Number Generator. In: Keller, A., Heinrich, S., Niederreiter, H. (eds.) Monte Carlo and Quasi-Monte Carl Methods 2006, pp. 607–622. Springer, Heidelberg (2008)
6. SIMD-oriented Fast Mersenne Twister (SFMT), `http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/SFMT/index-jp.html`
7. Johns, C.R., Brokenshire, D.A.: Introduction to the Cell Broadband Engine Architecture. IBM Journal of Research and Development 51, 503–520 (2007)

DNA Computing Hardware Design and Application to Multiclass Cancer Data

Sun-Wook Choi and Chong Ho Lee

School of Information and Communication Engineering
Inha University, Incheon 402-751, Korea
swchoi@inhaian.net, chlee@inha.ac.kr

Abstract. DNA computing-inspired pattern classification based on the hypernetwork model is a novel approach to pattern classification problems. The hypernetwork model has been shown to be a powerful tool for analysis of gene expression data. However, the ordinary hypernetwork model has limitations, such as using only binary data and operating sequentially. In this paper, we propose an improved method to process four-level data and to implement a hardware circuit for DNA computing-inspired pattern classifier. We show simulation results of multi-class cancer classification from the DNA microarray data for performance evaluation. Experiments show competitive diagnosis results over other conventional machine learning algorithms. Our four-level data approach also results stable and improved performance over the ordinary hypernetwork model.

1 Introduction

DNA computing, also known as molecular computing, is a new approach to massively parallel computation based on innovative work by Adleman. It has been applied to resolve problems in various areas [1]. In recent years, there has been novel approaches to pattern classification using the DNA computing-based molecular evolutionary learning model that uses only primitive DNA computing operations, such as hybridization and duplication, to automatically build a pattern classifier from a set of reference data [2]. However, implementation of DNA computing *in vitro* has certain difficulties. For example, the set of experimental constraints including temperatures, molecule densities, and salt concentrations, should be controlled strictly for precise simulation of *in vitro* reaction [3]. So, some alternative methods have been suggested to approximate the problem, such as probabilistic library model and hypernetwork model. These are simulated DNA computing approaches *in silico* [4].

DNA computing-inspired pattern classifier(DCPC) is based on the previous *in silico* approach using the hypernetwork model and its application to various areas, such as recognition of handwritten digits data and cancer classification, text classification, etc [5][6][7]. The DNA computing model shows competitive performance over other conventional algorithms, such as Support Vector Machine, k-Nearest Neighbor, Artificial Neural Network, etc. However, the ordinary models have some limitations. First of all, they are designed to use only

M. Köppen et al. (Eds.): ICONIP 2008, Part II, LNCS 5507, pp. 1072–1079, 2009.

binary data for processing. But, the binaryzation becomes cause of performance degradation because binarization process inevitably results in information loss. Secondly, using random masks in library creation process generates large variance of classification accuracy and causes unreliable diagnosis. Although the DNA computing approach has feature of massively parallel computation, the ordinary DNA computing models have been run on sequential machines.

In this paper, to overcome the above problems, we propose an improved method to process four-level data based on two data bits and implementing the DNA computing-inspired pattern classifier circuit on FPGA in order to provide massively parallel computation ability. We also show simulation results of multi-class cancer classification using DNA microarray data for analysis. It shows better performance than the ordinary method using binary data and has more competitive performance comparing with other algorithms. It also show stabilized classification accuracy and fast processing time.

2 DNA Computing-Based Pattern Classifier

2.1 Hypernetwork Model

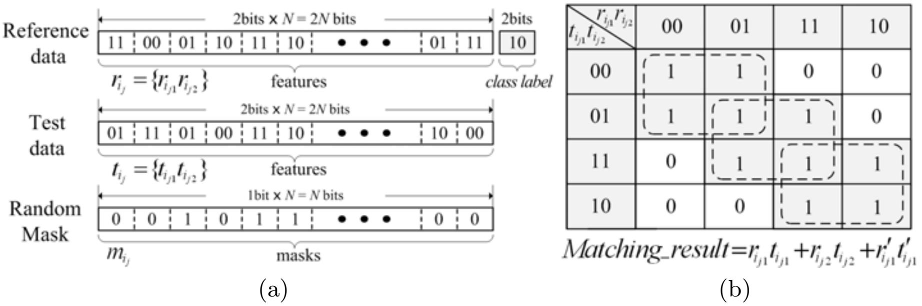
The DNA computing-inspired pattern classifier is based on the hypernetwork model [8]. The hypernetwork model has a random graph structure consisting of a large number of hyperedges. The hyperedges represent feature combinations and are sampled from the reference data. For each reference data, make d library elements of order k by random sampling the feature subsets of the reference data including class labels. The library of hypernetwork model includes multiple copies of each hyperegde and the number of copies means the importance of the hyperedge for the class.

For pattern classification problems, the model determines the closest classes by choosing the maximum matched hyperedge of the given test data. Here, the matching means that the values in a hyperedge are equal to the values of corresponding indices of the given test data. The classification of the hypernetwork model can be explained to the conditional probability of each class on the test data. Given test data x, the class $y*$ is predicted by computing the conditional probability of each class, and then selecting the class which has the highest conditional probability, as follows :

$$y* = \arg\max_{Y \in \{0,1,..,n\}} P(Y|x) = \arg\max_{Y \in \{0,1,..,n\}} \frac{P(Y,x)}{P(x)} \quad (1)$$

where $P(Y,x) = P(Y|x)P(x)$ and Y represents the candidate classes. More theoretical background of the hypernetwork model is found in [8]. The classification procedure is summarized as follows :

1. Present a test data x.
2. Extract all hyperedges that are matched to the test data x.
3. Count the number of each class in the extracted hyperedges.
4. Classify the given test data x as the class that counted most frequently.



$r_{i_{j1}}r_{i_{j2}}$ \ $t_{i_{j1}}t_{i_{j2}}$	00	01	11	10
00	1	1	0	0
01	1	1	1	0
11	0	1	1	1
10	0	0	1	1

$Matching\_result = r_{i_{j1}}t_{i_{j1}} + r_{i_{j2}}t_{i_{j2}} + r'_{i_{j1}}t'_{i_{j1}}$

Fig. 1. (a) Four-level data structure, (b) 2-bit matching table

2.2 Proposed Method

To overcome the limit of the previous DNA computing-based pattern classifier that use only binary data, we proposed a new method that is capable of using four-level data on the DNA computing-based pattern classifier. Before its description, we present new data structures on Fig. 1 (a). Let $\boldsymbol{R} = \{R_1, R_2, ..., R_n\}$ be the reference dataset. For k dimensionality of an input space, each reference data $R_i = \{r_{i_1}, r_{i_2}, ..., r_{i_k}, y_i\}$ consists of k components based on two bits feature where $r_{i_j} = \{r_{i_{j1}} r_{i_{j2}}\} \in \{00, 01, 11, 10\}$ and the class label $y \in \boldsymbol{Y}$. The new data structure will also apply equally to the test dataset $\boldsymbol{T} = \{T_1, T_2, ..., T_c\}$. To encode the four-level data with two bits, we use an empirical discretization method that described in section 4.

To determine whether the four-level data is matched or not, we propose a 2-bit matching table as shown in Fig. 1 (b). The rows and columns represent the feature variables of reference data and test data. The value on intersection of a column and a row means that whether corresponding feature variable is matched or not. For example, let one feature of reference data is '01' and the corresponding feature of test data is '01'. In this case, the value '1' on intersection of column '01' and row '01' means matched. One the other way, let one feature of reference data is '01' and corresponding feature of test data is '10', the value '0' on the intersection of column '01' and row '10' means mismatched.

And also, the proposed matching table allows that the adjacent data is matched. For example, the values '00' and '01' are not same but regarded as matched because the two are adjacent. One benefit of the proposed 2-bit matching table is obtaining an effect of improving noise tolerance. As a result, we acquire the improved performance with robust to approximation. The matching table is simply represented using boolean algebra as follows:

$$Matching\ result = r_{i_{j1}} t_{i_{j1}} + r_{i_{j2}} t_{i_{j2}} + {r_{i_{j1}}}' {t_{i_{j1}}}' \qquad (2)$$

And a modified matching comparator can be implemented simply based on equation (2) using AND, OR, NOT gates as shown in Fig. 4.

3 Implementation of Hardware Architecture

3.1 System Overview

The hardware architecture, which is composed of the test data register, control block, processing block(consist of the reference memory, random mask generator, compare block, summation block) and maxfit block, is shown in Fig. 2. The processing blocks are operated separately for reference data of each class. And the parallel operation in each compare block generates high throughput. The extension of the processing block is also possible, when other dataset with numerous classes are given to classify.

Operating phases of the hardware consists of loading, comparing, counting and prediction. In the loading and comparing phase, the test data is given to each compare block concurrently through a common bus with the reference data and random masks of each class. And the matched results from each compare block are counted in the summation block. At last, the most frequently counted class is chosen as the classification result out from the maxfit block.

3.2 Building Blocks

As shown in Fig. 2, this system has N memories storing the reference data for each class. The reference data are compared with given test data using the proposed 2-bit comparator(see Fig. 4) in the compare block. As mentioned in section 2.2, if the 2-bit comparator has adjacent input values, the output result is ‘1’ otherwise, the output result is ‘0’. These compare operations processed massive-parallel on the compare block. For example, if the reference data has N

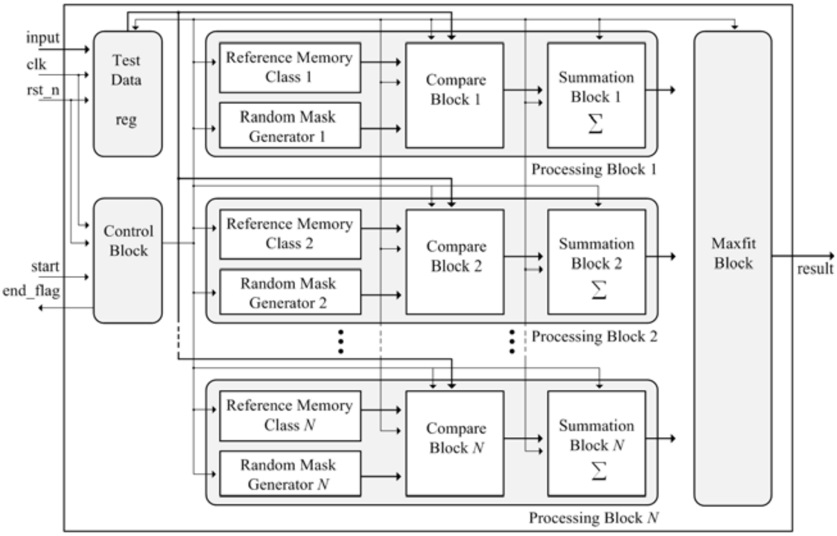


Fig. 2. The hardware scheme

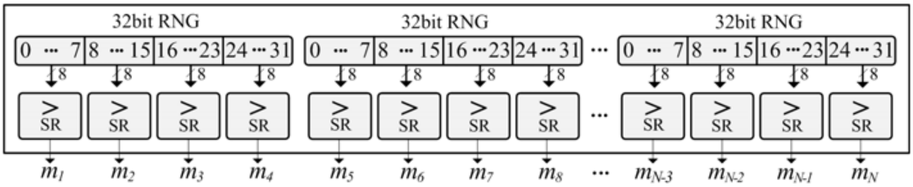


Fig. 3. Random Mask Generator

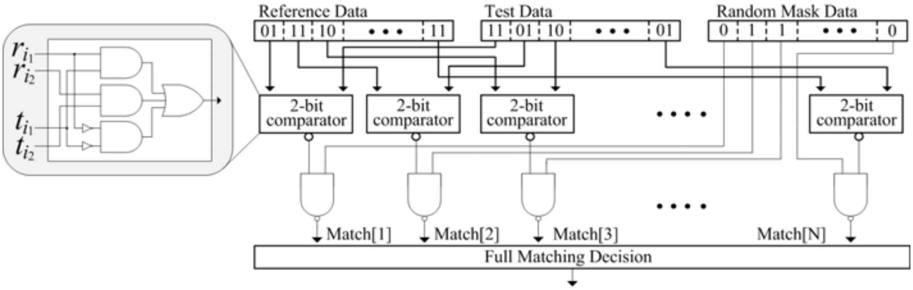


Fig. 4. Compare Block

classes and M feature variables per data, then $M \times N$ compare operations are processed concurrently. After then, the compared output results are randomly selected using random mask data created by a random mask generator(RMG) as shown in Fig. 3. The RMG was designed in order to select the proper number of order using preset selection ratio. The RMG is composed of 32-bit random number generator(RNG) that is based on a combination of linear feedback shift register(LFSR) and a cellular automata shift register(CASR) as presented in [9], and each created random number has value in eight bits width (0 to 255). For example, 30 of 32-bit RNG is needed to create random masks of 120 bits because one RNG creates four random mask bits. Using the proposed RMG, the system has some advantages, such as reduction of RAM usage and generalization capacity to change data, in comparison with the previous studies [6][10].

When all of the inverted results from the 2-bit comparators and mask values of corresponding indices of the random mask data are assigned to each NAND gate, if all outputs of the NAND gates are the same value of 1, it means that the given test data and the hyperedge are matched. The counted matched results from each summation block are used in order to predict the classification result on the maxfit block. The comparing and counting operation on the processing block are repeated as the multiplied number by the number of reference data and created random masks for each class.

Processes of the above building blocks emulate effectively the massively parallel molecular operation of the DNA computing-inspired pattern classification *in vitro*.

Table 1. Multi-class cancer datasets

Dataset	No. of classes	No. of genes	No. of samples		No. of selected gene
			Reference	Test	
Subtype of ALL	6	12558	163	85	90
SRBCT	4	2308	63	20	120
Lung cancer	5	12600	136	67	150
Leukemia	3	7129	38	34	90

Table 2. Diagnostic accuracy

Dataset	DNAC-inspired pattern classifier(%)		SVM(%)	ANN(%)	kNN(%)
	binary(SD)	four level(SD)			
Subtype of ALL	93.57(1.48)	98.34(0.67)	98.00	98.50	97.16
SRBCT	93.80(3.70)	100.0(0.00)	100.0	91.03	86.90
Lung cancer	89.31(1.47)	94.03(0.00)	96.05	87.80	89.64
Leukemia	94.76(2.11)	98.58(1.69)	97.50	76.61	83.57
Average	92.86(2.19)	97.74(0.59)	97.88	88.49	89.32

4 Experiments

4.1 Multiclass Cancer Classification

To evaluate the usefulness of the proposed method, we carries out experiments on the following four public datasets of microarray data for the multiclass cancer classification. The subtypes of ALL dataset contains six subtypes of acute lymphoblastic leukemia(ALL) [11]. The small round blue cell tumors(SRBCT) dataset contains four different childhood tumors [12]. The lung cancer dataset contains four subtypes of lung cancer and normal sample [13]. The leukemia dataset contains three subtypes of leukemia [14]. Above four datasets are summarized in Table 1.

4.2 Data Preprocessing and Experimental Setup

As shown in Table 1, large number of genes together with relatively small number of samples is one characteristic of gene expression data available for cancer classification problems. Such characteristic induces that gene selection is a necessary procedure for cancer classification with gene expression data to ensure reliable and meaningful classification results along with other benefits such less data storage and computation cost. In our experiments, we use the signal-to-noise statistic $(\mu_0 - \mu_1)/(\sigma_0 + \sigma_1)$, where μ and σ represent the mean and standard deviation of gene expression values, respectively, for each class [14]. The gene selection is performed based on the reference data to avoid overfitting. Each sample is normalized to have standard distribution before being processed.

In addition, it is necessary to discretize the gene expression value for using the DNA computing-based pattern classifier. We determine a threshold using mean of the whole genes that belongs to the same sample for the previous DNA computing model that uses binary data. And we determine three threshold values (high, mid, low) by experiments in order to apply the proposed method that uses four-level data. Using three threshold values, we discretize the gene expression value into four-level data (00, 01, 11, 10).

To assess the performance of classifiers, the partition of datasets used in the original papers into training and test data is used in our experiment. And we figure out the optimum number of order and random masks of classifiers for each dataset by experiment. After that, we measure the accuracy of classification for each dataset. The accuracy of classification is averaged of 100 iterations. The results of previous studies [11][12][13][14] are compared to assess the performance of classifications.

4.3 Results

Table 2 shows the comparison of the classification accuracies of the proposed method and previously published studies using other conventional machine learning methods. It shows the difference between the standard deviations of the previous methods and the proposed method. The proposed method shows 97.74% of average accuracy. It is better than the ordinary DNA computing model, the ANN, and the k-NN, while providing competitive or better performance over the SVM. Also The proposed method shows lower standard deviation(SD) than the previous DNA computing model. It means that the proposed method has more stable performance.

4.4 Implementation on FPGA

The hardware is applied to the SRBCT dataset. The DNA computing-inspired pattern classifier for the SRBCT datset is designed with VerilogHDL and implemented on Xilinx Virtex-4 LX FPGA XC4VLX200. The implemented hardware contains 480 of 2-bit comparator, 4 of RMG each has 30 of 32-bit RNG, 4 of reference data RAM each is 242 bits wide. When synthesized for target device, the whole system uses 11511 slices that is 12% of the device capacity. The implemented hardware runs at 100 MHz with throughput of 3.62×10^4 data/s for classfication process. The number of clock cycles needed for one data classification is 2760. And it takes nearly 27.6 micro seconds to process one data.

5 Conclusion

In order to overcome the limits of the ordinary DNA computing model that uses only binary data, we propose the improved method which utilizes four-level expression data. The proposed method is applied to four cases of multi-class cancer classifications. The results show that our method outperforms conventional machine learning algorithms in terms of diagnostic accuracy. It also produces more stable and improved performance than the previously developed models of DNA Computing. We also propose the new hardware architecture of DNA computing-inspired pattern classifier, designed on XILINX Virtex-4 LX platform, which is suitable for multi-class cancer classifications. The speed of this hardware is fast enough to comply point-of-care diagnostic purpose.

References

1. Adleman, L.: Molecular computation of solutions to combinatorial problems. Science 266(5187), 1021–1024 (1994)
2. Zhang, B.-T., Jang, H.-Y.: A bayesian algorithm for in vitro molecular evolution of pattern classifiers. In: Ferretti, C., Mauri, G., Zandron, C. (eds.) DNA 2004. LNCS, vol. 3384, pp. 458–467. Springer, Heidelberg (2005)
3. Wetmur, J.: Physical chemistry of nucleic acid hybridization. DNA Based Computers III, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 48, 1–23 (1999)
4. Zhang, B.T., Kim, J.K.: Dna hypernetworks for information storage and retrieval. In: Mao, C., Yokomori, T. (eds.) DNA12. LNCS, vol. 4287, pp. 298–307. Springer, Heidelberg (2006)
5. Ha, J., Eom, J., Kim, S., Zhang, B.: Evolutionary hypernetwork models for aptamer-based cardiovascular disease diagnosis. In: Proc. of the GECCO 2007, pp. 2709–2716. ACM, New York (2007)
6. Kim, B.S., Kim, J.K., Kwon, O.H., Hwang, S.K., Song, J.Y., Zhang, B.T., Lee, C.H., Park, J., Chung, D.J.: Hardware implementation of the pattern recognize processor base on dna computing technique. In: Proc. of The 22nd ITC-CSCC (2007)
7. Kwon, O.H., Wang, K.Y., Kim, J.Y., Park, J., Chung, D.J., Lee, C.H.: Dna inspired digital signal pattern matching algorithm. In: Proc. of FBIT 2007, pp. 753–756 (2007)
8. Zhang, B.T.: Hypernetworks: A molecular evolutionary architecture for cognitive learning and memory. Computational Intelligence Magazine, IEEE 3(3), 49–63 (2008)
9. Tkacik, T.: A hardware random number generator. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 450–453. Springer, Heidelberg (2003)
10. Kim, J.K., Zhang, B.T.: Evolving hypernetworks for pattern classification. In: Proc. of IEEE CEC 2007, pp. 1856–1862 (2007)
11. Yeoh, E., Ross, M., Shurtleff, S., Williams, W., Patel, D., Mahfouz, R., Behm, F., Raimondi, S., Relling, M., Patel, A., et al.: Classification, subtype discovery, and prediction of outcome in pediatric acute lymphoblastic leukemia by gene expression profiling. Cancer Cell 1(2), 133–143 (2002)
12. Khan, J., Wei, J., Ringnér, M., Saal, L., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C., Peterson, C., et al.: Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. Nature Medicine 7, 673–679 (2001)
13. Bhattacharjee, A., Richards, W., Staunton, J., Li, C., Monti, S., Vasa, P., Ladd, C., Beheshti, J., Bueno, R., Gillette, M., et al.: Classification of human lung carcinomas by mrna expression profiling reveals distinct adenocarcinoma subclasses. In: Proceedings of the National Academy of Sciences, pp. 13790–13795 (2001)
14. Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M., et al.: Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. Science 286(5439), 531–537 (1999)

Author Index