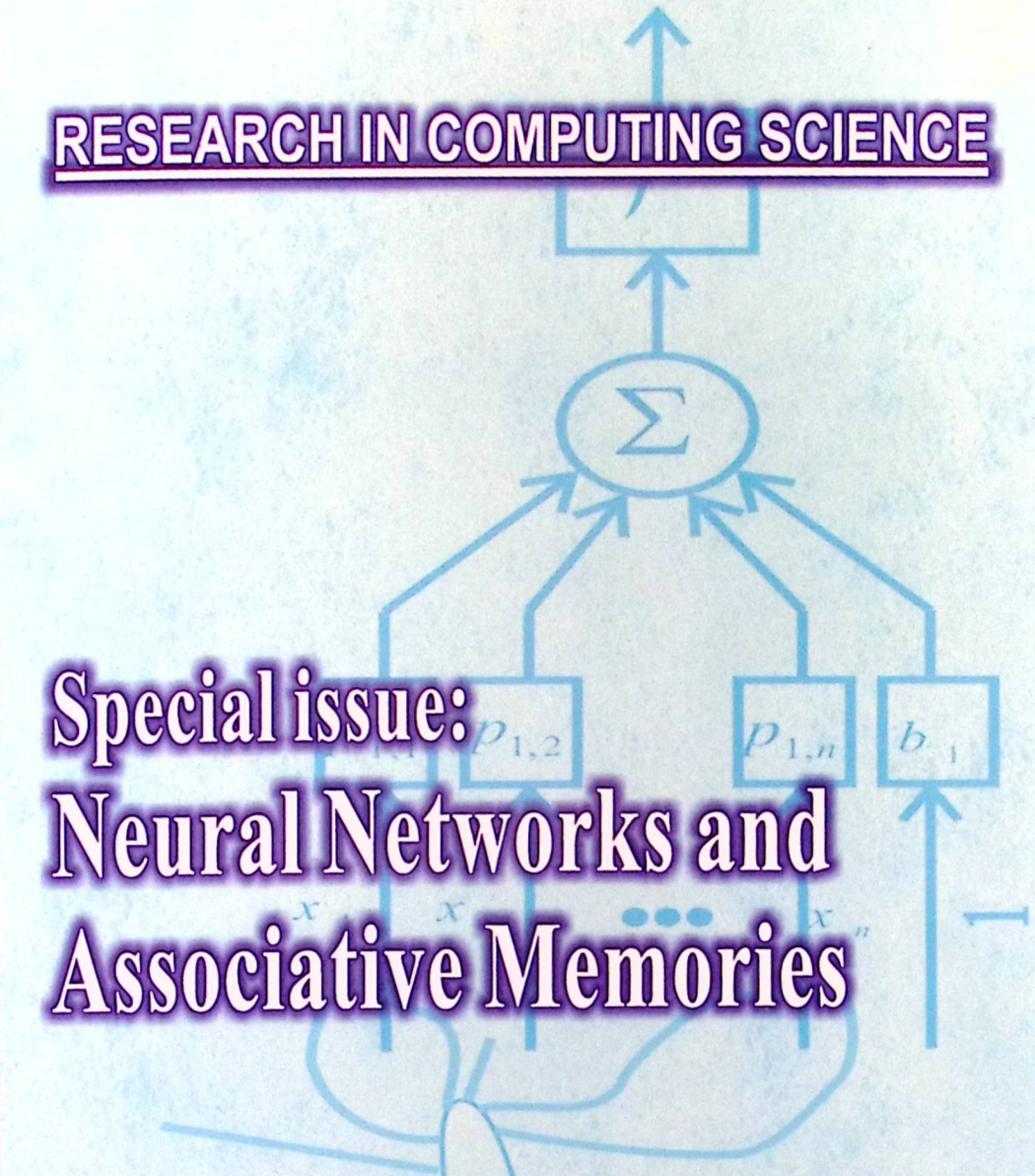
RESEARCH IN COMPUTING SCIENCE

ISSN: 1870-4069



Special issue: Neural Networks and Associative Memories

Juan Humberto Sossa Azuela
Ricardo Barrón Fernández
Edgardo Manuel Felipe Riverón
(Eds.)

Vol. 28

RCS
Research in Computing Science

Special Issue in Neural Networks And Associative Memories

Research in Computing Science

Series Editorial Board

Comité Editorial de la Serie

Editors-in-Chief:

Editores en Jefe

Juan Humberto Sossa Azuela (Mexico)
Gerhard Ritter (USA)
Jean Serra (France)
Ulises Cortés (Spain)

Associate Editors:

Editores Asociados

Jesús Angulo (Frane)
Jihad El-Sana (Israel)
Jesús Figueroa (Mexico)
Alexander Gelbukh (Russia)
Ioannis Kakadiaris (USA)
Serguei Levachkine (Russia)
Petros Maragos (Greece)
Julian Padget (UK)
Mateo Valero (Spain)

Editorial Coordination:

Coordinación Editorial

Blanca Miranda Valencia

Formatting:

Formación

J. Humberto Sossa Azuela

Research in Computing Science es una publicación trimestral, de circulación internacional, editada por el Centro de Investigación en Computación del IPN, para dar a conocer los avances de investigación científica y desarrollo tecnológico de la comunidad científica internacional. **Volumen 28**, Noviembre, 2007. Tiraje: 500 ejemplares. *Certificado de Reserva de Derechos al Uso Exclusivo del Título* No. 04-2004-062613250000-102, expedido por el Instituto Nacional de Derecho de Autor. *Certificado de Licitud de Título* No. 12897, *Certificado de licitud de Contenido* No. 10470, expedidos por la Comisión Calificadora de Publicaciones y Revistas Ilustradas. El contenido de los artículos es responsabilidad exclusiva de sus respectivos autores. Queda prohibida la reproducción total o parcial, por cualquier medio, sin el permiso expreso del editor, excepto para uso personal o de estudio haciendo cita explícita en la primera página de cada documento. Impreso en la Ciudad de México, en los Talleres Gráficos del IPN – Dirección de Publicaciones, Tres Guerras 27, Centro Histórico, México, D.F. Distribuida por el Centro de Investigación en Computación, Av. Juan de Dios Bátiz S/N, Esq. Av. Miguel Othón de Mendizábal, Col. Nueva Industrial Vallejo, C.P. 07738, México, D.F. Tel. 57 29 60 00, ext. 56571.

Editor Responsable: *Juan Humberto Sossa Azuela,* RFC *SOAJ560723*

***Research in Computing Science** is published by the Center for Computing Research of IPN. **Volume 28**, November, 2007. Printing 500. Authors are responsible for the contents of their papers. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without prior permission of Centre for Computing Research. Printed in Mexico City, November, 2007, in the IPN Graphic Workshop – Publication Office.*

Volume 28

Volumen 28

Special Issue in Neural Networks and Associative Memories

Volume Editors:
Editores del Volumen

Juan Humberto Sossa Azuela
Ricardo Barrón Fernández
Edgardo Manuel Felipe Riverón

Instituto Politécnico Nacional
Centro de Investigación en Computación
México 2007

ISSN: 1870-4069

Copyright © Instituto Politécnico Nacional 2007
Copyright © Instituto Politécnico Nacional 2007

Instituto Politécnico Nacional (IPN)
Centro de Investigación en Computación (CIC)
Av. Juan de Dios Bátiz s/n esq. M. Othón de Mendizábal
Unidad Profesional "Adolfo López Mateos", Zacatenco
07738, México D.F., México

http.//www.ipn.mx
http://www.cic.ipn.mx

The editors and the Publisher of this journal have made their best effort in preparing this special issue, but make no warranty of any kind, expressed or implied, with regard to the information contained in this volume.

All rights reserved. No part of this publication may be reproduced, stored on a retrieval system or transmitted, in any form or by any means, including electronic, mechanical, photocopying, recording, or otherwise, without prior permission of the Instituto Politécnico Nacional, except for personal or classroom use provided that copies bear the full citation notice provided on the first page of each paper.

Indexed in LATINDEX and PERIODICA
Indexada en LATINDEX y PERIODICA

Printing: 500
Tiraje: 500

Printed in Mexico
Impreso en México

Preface

Neural Networks (NN) and Associative Memories (AM) are tools that have been used for many years. Their theoretical foundations are now well established. Their applications are numerous.

This special issue presents original research papers on the internal art and craft of Neural Networks and Associative Memories: theoretical foundations, specific methodologies and real world applications. This special volume is structured into seven thematic fields representative of some of the main current areas of interest within the NN and AM community:

- Feature Extraction and Dimensionality Reduction,
- Image Processing and Applications,
- Neural Networks and Associative Memories,
- Real World Applications,

A total of 35 full papers from 8 different countries were received for evaluation. Each submission was reviewed by at least two independent members of the Editorial Board of the volume. This volume contains revised versions of 20 papers, selected for publication after a thorough evaluation. Thus the acceptance rate was of 57%. The volume also includes the invited paper: A General Framework for Dynamic Networks by Professor Martin Hagan and Orlando de Jesús.

We cordially thank all people involved in the preparation of this volume. In the first place there are the authors of the papers constituting it; it is the excellence of their research work that gives sense to the work of all other people involved. We thank the members of the Editorial Board of the volume and additional referees. The submission, reviewing and selection process was supported for free by the EasyChair system, www.EasyChair.org.

Humberto Sossa
Ricardo Barrón
Edgardo Felipe

November 2007

Table of Contents
Índice

Page/Pág.

Invited Paper

A General Framework for Dynamic Networks 3
Martin Hagan and Orlando De Jesús

Neural Networks and Associative Memories

Infinite Dimensional Associative Memories: Foundations (with Theorems) 15
Enrique Carlos Segura

Genetically Modified GMDH Method with Cloning 29
Marcel Jiřina, Marcel Jiřina, jr.

Polynomial Cellular Neural Networks for Implementing Semitotalistic Cellular Automata 39
Giovanni Egidio Pazienza, Eduardo Gomez-Ramírez and Xavier Vilasís-Cardona

Chaos in a Single Recurrent Artificial Neuron..... 49
L. Torres-Treviño

Image Processing Applications

Image Retrieval under Image Transformations and Occlusions 59
Roberto A. Vázquez and Humberto Sossa

Cursive Character Recognition by combining Describing Features, Slalom Method and Daubechies Wavelet 69
L. K. Toscano, J. H. Sossa, R. Barron, G. Sanchez

Cluster Tree Self Organizing Map for Developing Image Retrieval System 81
Hamed Shahbazi, Mohsen Soryani, Nasser Mozayani and Mahmood Fathy

Feature Extraction and Dimensionality Reduction

Using Testor Theory to Reduce the Dimension of Neural Network Models 93
Roberto A. Vázquez and Salvador Godoy-Calderón

Feature Extraction using Gabor and DWT for GMM based Face Verification Algorithms 105
Jesus Olivares-Mercado, Gabriel Sanchez-Perez, Mariko Nakano-Miyatake, Hector Perez-Meana

Perceptron for Feature Selection 115
Manuel Mejía-Lavalle and Enrique Sucar

A Modified Bottleneck Neural Network for Dimensionality Reduction..... 127
Eduardo Filemón Vázquez Santacruz, Debrup Chakraborty

Real World Applications

Multiple Fault Diagnosis in Electrical Power Systems with Dynamic Load Changes using Probabilistic Neural Networks..... 139
Juan Pablo Nieto González, Luis E. Garza Castañon, Rubén Morales Menendez

Improving Under Sampling with Neural Networks for Class Imbalance Problem 151
Man-Sun Kim

Closing Price Prediction for Auctions of Hotel Rooms on the TAC Classic **161**
Eber Jair Flores Andonegui, Darnes Vilariño Ayala, Fabiola López y López

Reinforcement Learning based Neurocontrollers **173**
Erik Cuevas, Daniel Zaldivar, Marco Perez and Raúl Rojas

Direct Adaptive Vector Neural Control of a Three Phase Induction Motor **183**
Ieroham S. Baruch, Carlos R. Mariaca-Gaspar and Irving P. de la Cruz

Fault Diagnosis in Power Transmission Networks using Bayesian Networks **193**
Luis E. Garza and Geovanna Ruffo

An Indirect Adaptive Neural Control of a Wastewater Treatment Bioprocess via Marquardt Learning **203**
Carlos R. Mariaca-Gaspar, and Ieroham S. Baruch

The Simulation Model of Highpressure, Suspension Waterjet Cutting Process of Marble **215**
Andrzej Perec

Induction Learning Techniques Applied to Bayesian Networks Optimization **225**
P. Britos, P. Felgaer, D. Rodríguez and R. García-Martínez

Invited Paper

A General Framework for Dynamic Networks

Martin Hagan[1] and Orlando De Jesús[2]

[1] School of Electrical and Computer Engineering, Oklahoma State University,
Stillwater, Oklahoma, 74078
mhagan@ieee.org
[2] Halliburton - Carrollton Technology Center - Research,
Halliburton Energy Services
Carollton, Texas, 75006
Orlando.DeJesus@Halliburton.com

(*Invited Paper*)

Abstract. The field of dynamic neural networks is expansive. It has applications in such disparate areas as control systems, prediction in financial markets, channel equalization in communication systems, phase detection in power systems, sorting, fault detection, speech recognition, and even the prediction of protein structure in genetics. Within these various application areas, a great number of dynamic neural network architectures have been proposed. These dynamic networks are often trained using gradient-based optimization algorithms. In this paper we will attempt to present a unified view of the training of dynamic networks. We will begin with a very general framework for representing dynamic networks and will demonstrate how gradient-based algorithms can be efficiently developed using this framework.

1 Introduction

Dynamic networks are networks that contain delays (or integrators, for continuous-time networks). These dynamic networks can have purely feedforward connections, or they can also have some feedback (recurrent) connections. Dynamic networks have memory. Their response at any given time will depend not only on the current input, but on the history of the input sequence.

Because dynamic networks have memory, they can be trained to learn sequential or time-varying patterns. This has applications in such diverse areas as control of dynamic systems [1], prediction in financial markets[2], channel equalization in communication systems [3], phase detection in power systems [4], sorting [5], fault detection [6], speech recognition [7], learning of grammars in natural languages [8], and even the prediction of protein structure in genetics [9].

Dynamic networks can be trained using standard gradient-based or Jacobian-based optimization methods. However, the gradients and Jacobians that are required for these methods cannot be computed using the standard backpropagation algorithm. In this paper we will discuss a general dynamic network framework, in which dynamic backpropagation algorithms can be efficiently developed.

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 3-12

There are two general approaches (with many variations) to gradient and Jacobian calculations in dynamic networks: backpropagation-through-time (BPTT) [10] and real-time recurrent learning (RTRL) [11]. In the BPTT algorithm, the network response is computed for all time points, and then the gradient is computed by starting at the last time point and working backwards in time. This algorithm is computationally efficient for the gradient calculation, but it is difficult to implement on-line, because the algorithm works backward in time from the last time step.

In the RTRL algorithm, the gradient can be computed at the same time as the network response, since it is computed by starting at the first time point, and then working forward through time. RTRL requires more calculations than BPTT for calculating the gradient, but RTRL allows a convenient framework for on-line implementation. For Jacobian calculations, the RTRL algorithm is generally more efficient than the BPTT algorithm [12,13].

In order to more easily present general BPTT and RTRL algorithms, it will be helpful to introduce modified notation for networks that can have recurrent connections. In the next section we will introduce this notation and will develop a general dynamic network framework. The following section will present procedures for computing gradients for the general framework.

The RTRL and BPTT methods can be thought of as general concepts, rather than as specific algorithms that can be implemented as general computer codes applicable to arbitrary network architectures. The RTRL gradient algorithm has been discussed in a number of previous papers [11, 14], but generally in the context of specific network architectures. The BPTT gradient algorithm has been described as a basic concept in [10], and a diagrammatic method for deriving the gradient algorithm for a certain class of architectures has been provided in [15].

As a general rule, there have been two major approaches to using dynamic training. The first approach has been to use the general RTRL or BPTT concepts to derive algorithms for particular network architectures. The second approach has been to put a given network architecture into a particular canonical form (e.g., [16-18]), and then to use the dynamic training algorithm which has been previously designed for the canonical form.

In this paper our approach will be to develop a very general framework in which to conveniently represent a large class of dynamic networks, and then to derive the RTRL and BPTT algorithms for the general framework. In this way, one computer code can be used to train arbitrarily constructed network architectures, without requiring that each architecture be first converted to a particular canonical form.

2 Development of a General Class of Dynamic Network

We will build up to our general notation by starting with a simple multilayer perceptron network, as shown in Fig. 1. The equations of operation for this network are

$$\mathbf{n}^{m+1} = \mathbf{W}^{m+1}\mathbf{a}^{m} + \mathbf{b}^{m+1} \quad \text{for } m = 0,1,\ldots,M-1 \tag{1}$$

$$\mathbf{a}^m = \mathbf{f}^m\left(\mathbf{n}^m\right) \tag{2}$$

where $\mathbf{n}^m$ is the net input at layer m, $\mathbf{a}^m$ is the output of layer m, and $\mathbf{a}^0 = \mathbf{p}$ is the input to the network. The overall network output is the output of the last layer, $\mathbf{a}^M$.

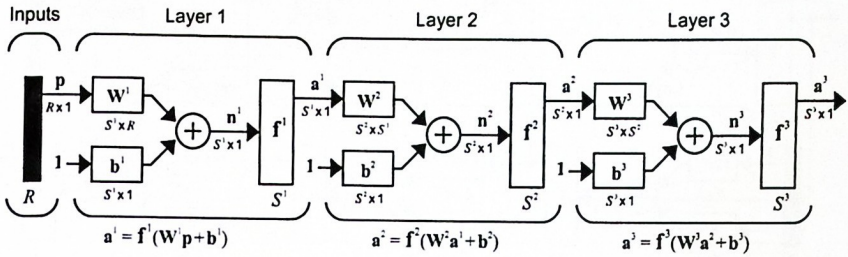


Fig. 1. Multilayer Perceptron

In the Multilayer Perceptron, each layer connects forward to the following layer, and each layer consists of four components:

1. A weight matrix $\mathbf{W}^m$,
2. A bias vector $\mathbf{b}^m$,
3. A summing junction, and
4. A transfer function $\mathbf{f}^m(\ )$.

We can generalize the Multilayer Perceptron by allowing layers to connect forward to an arbitrary number of other layers. In addition, we can have multiple input vectors, each one of which can be connected to any layer. We call the resulting class of networks Layered Feedforward Neural Networks (LFNN). An example is shown in Fig. 2. The equations of operation of an LFFN can be written as

$$\mathbf{n}^m(t) = \sum_{l \in I_m} \mathbf{IW}^{m,l}\mathbf{p}^l(t) + \sum_{l \in L_f^m} \mathbf{LW}^{m,l}\mathbf{a}^l(t) + \mathbf{b}^m \tag{3}$$

$$\mathbf{a}^m = \mathbf{f}^m\left(\mathbf{n}^m\right) \tag{4}$$

where I_m is the set of indices of all inputs that connect to layer m, L_f^m is the set of indices of all layers that connect forward to layer m., $\mathbf{p}^l$ is the l^{th} input to the network, $\mathbf{IW}^{m,l}$ is the *input* weight between input l and layer m, $\mathbf{LW}^{m,l}$ is the *layer* weight between layer l and layer m, and $\mathbf{b}^m$ is the bias vector for layer m.

For the LFFN class of networks, we can have multiple weight matrices associated with each layer - some coming from external inputs, and others coming from other layers.

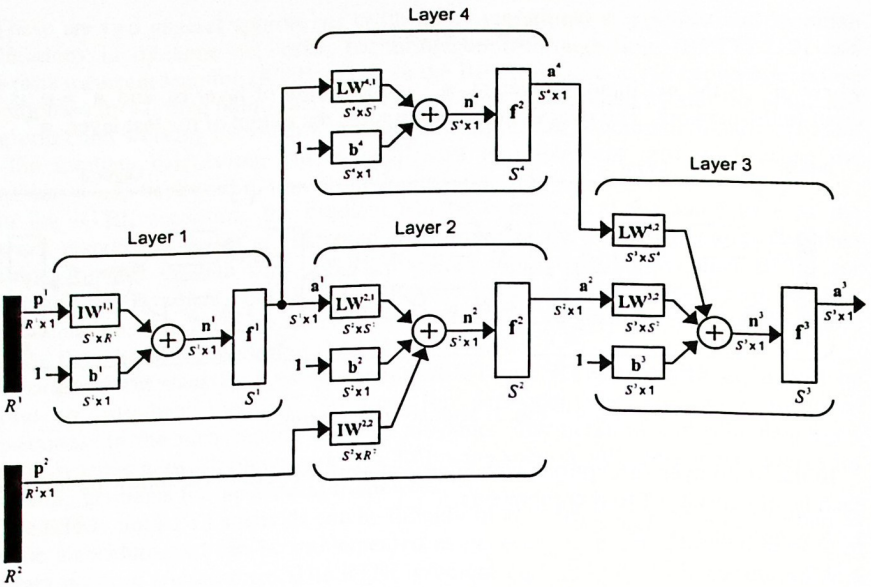


Fig. 2. Example Layered Feedforward Network

So far, we have considered only static networks. The LFFN class of networks can be further generalized to allow delays to be associated with each weight matrix. This leads to Layered Digital Dynamic Networks (LDDN) [12]: the fundamental unit of this framework is the layer (as with the multilayer perceptron); the networks are digital as opposed to analog (or discrete-time as opposed to continuous-time); and we use the term "dynamic" rather than "recurrent" because we want to include feedforward networks that have memory.

An example of a dynamic network in the LDDN framework is shown in Fig. 3. Note that the key component that has been added to certain layers is the tapped delay line (indicated by TDL in the figure). The equations of operation of an LDDN network are

$$\mathbf{n}^m(t) = \sum_{l \in I_m} \sum_{d \in DI_{m,l}} \mathbf{IW}^{m,l}(d)\mathbf{p}^l(t-d) + \sum_{l \in L_f^m} \sum_{d \in DL_{m,l}} \mathbf{LW}^{m,l}(d)\mathbf{a}^l(t-d) + \mathbf{b}^m \tag{5}$$

$$\mathbf{a}^m(t) = \mathbf{f}^m\left(\mathbf{n}^m(t)\right) \tag{6}$$

where $DL_{m,l}$ is the set of all delays in the tapped delay line between Layer l and Layer m, $DI_{m,l}$ is the set of all delays in the tapped delay line between Input l and Layer m

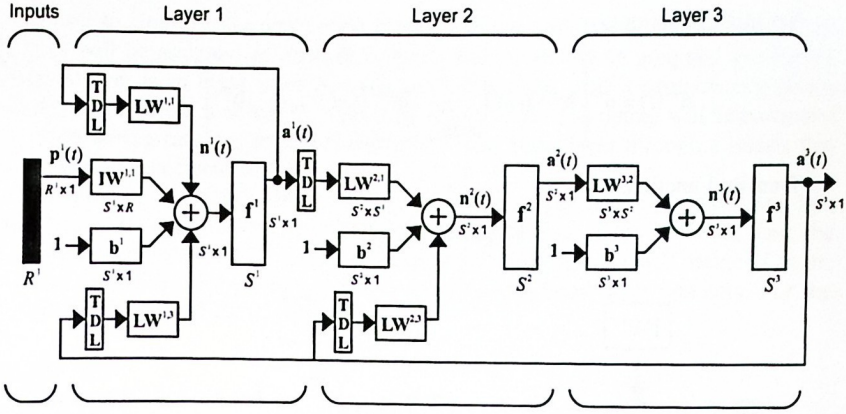


Fig. 3. Example Dynamic Network in the LDDN Framework

The LDDN framework is quite general. It is equivalent to the class of general ordered networks discussed in [10] and [19]. It is also equivalent to the signal flow graph class of networks used in [15] and [20]. However, we can increase the generality of the LDDN further. In all of the classes of networks that we have presented so far, the weight matrix multiplies the corresponding vector coming into the layer (from an external input in the case of **IW**, and from another layer in the case of **LW**). This means that a dot product is formed between each row of the weight matrix and the input vector.

We can consider more general *weight functions* than simply the dot product. For example, radial basis layers compute the distances between the input vector and the rows of the weight matrix. We can allow weight functions with arbitrary (but differentiable) operations between the weight matrix and the input vector. This enables us to include higher-order networks as part of our framework.

Another generality we can introduce is for the *net input function*. This is the function that combines the results of the weight function operations with the bias vector. In all of the networks that we have considered to this point, the net input function has been a simple summation. We can allow arbitrary, differentiable net input functions to be used.

The resulting network framework is the Generalized LDDN (GLDDN). A block diagram for a simple GLDDN (without delays) is shown in Fig. 4. The equations of operation for a GLDDN are

Weight Functions:

$$\mathbf{iz}^{m,l}(t,d) = \mathbf{ih}^{m,l}\left(\mathbf{IW}^{m,l}(d), \mathbf{p}^{l}(t-d)\right) \tag{7}$$

$$\mathbf{lz}^{m,l}(t,d) = \mathbf{lh}^{m,l}\left(\mathbf{LW}^{m,l}(d), \mathbf{a}^{l}(t-d)\right) \tag{8}$$

Net Input Functions:

$$\mathbf{n}^m(t) = \mathbf{o}^m\left(\mathbf{iz}^{m,l}(t,d)\Big|_{\substack{l\in I_m \\ d\in DI_{m,l}}}, \mathbf{lz}^{m,l}(t,d)\Big|_{\substack{l\in L_m^f \\ d\in DL_{m,l}}}, \mathbf{b}^m\right) \tag{9}$$

Transfer Functions:

$$\mathbf{a}^m(t) = \mathbf{f}^m\left(\mathbf{n}^m(t)\right) \tag{10}$$

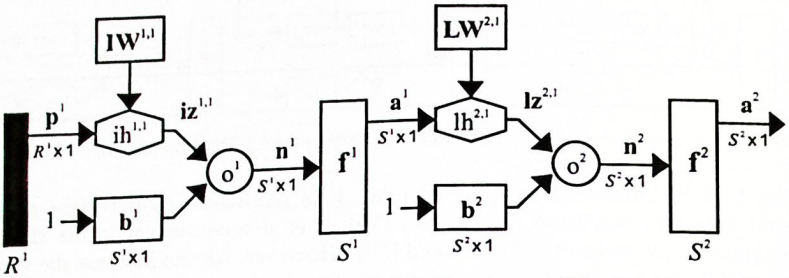


Fig. 4. Example Network with General Weight Functions and Net Input Functions

3 Gradient Calculation for the GLDDN

The next step is to develop an algorithm for computing the gradient for the GLDDN. This can be done using the BPTT or the RTRL approaches. Because of limited space, we will describe only the RTRL algorithm in this paper. (Both approaches are described for the LDDN framework in [12].)

To explain the gradient calculation for the GLDDN, we must create certain definitions. We do that in the following paragraphs.

3.1 Preliminary Definitions

First, as we stated earlier, a *layer* consists of a set of *weights*, associated *weight functions*, associated *tapped delay lines*, a *net input function*, and a *transfer function*. The network has *inputs* that are connected to special weights, called *input weights*. The weights connecting one layer to another are called *layer weights*. In order to calculate the network response in stages, layer by layer, we need to proceed in the proper layer order, so that the necessary inputs at each layer will be available. This ordering of layers is called the *simulation order*. In order to backpropagate the derivatives for the gradient calculations, we must proceed in the opposite order, which is called the *backpropagation order*.

In order to simplify the description of the gradient calculation, some layers of the GLDDN will be assigned as network outputs, and some will be assigned as network inputs. A layer is an *input layer* if it has an input weight, or if it contains any delays with any of its weight matrices. A layer is an *output layer* if its output will be compared to a target during training, or if it is connected to an input layer through a matrix that has any delays associated with it.

For example, the LDDN shown in Fig. 3 has two output layers (1 and 3) and two input layers (1 and 2). For this network the simulation order is 1-2-3, and the backpropagation order is 3-2-1. As an aid in later derivations, we will define U as the set of all output layer numbers and X as the set of all input layer numbers. For the LDDN in Fig. 3, $U=\{1,3\}$ and $X=\{1,2\}$.

3.2 Gradient Calculation

The objective of training is to optimize the network performance, quantified in the performance index $F(\mathbf{x})$, where $\mathbf{x}$ is a vector containing all of the weights and biases in the network. In this paper we will consider gradient-based algorithms for optimizing the performance (e.g., steepest descent, conjugate gradient, quasi-Newton, etc.). For the RTRL approach, the gradient is computed using

$$\frac{\partial F(\mathbf{x})}{\partial \mathbf{x}} = \sum_{t=1}^{Q} \sum_{u \in U} \left[\frac{\partial \mathbf{a}^u(t)}{\partial \mathbf{x}^T} \right]^T \frac{\partial^e F(\mathbf{x})}{\partial \mathbf{a}^u(t)}, \tag{11}$$

where

$$\frac{\partial \mathbf{a}^u(t)}{\partial \mathbf{x}^T} = \frac{\partial^e \mathbf{a}^u(t)}{\partial \mathbf{x}^T} + \sum_{u' \in U} \sum_{x \in X} \sum_{d \in DL_{x,u'}} \frac{\partial^e \mathbf{a}^u(t)}{\partial \mathbf{n}^x(t)^T} \frac{\partial^e \mathbf{n}^x(t)}{\partial \mathbf{a}^{u'}(t-d)^T} \frac{\partial \mathbf{a}^{u'}(t-d)}{\partial \mathbf{x}^T}. \tag{12}$$

The superscript e in these expressions indicates an explicit derivative, not accounting for indirect effects through time.

Many of the terms in Eq. 12 will be zero and need not be included. To take advantage of these efficiencies, we introduce the following definitions

$$E_{LW}^{U}(x) = \left\{ u \in U \ni \exists \left(\mathbf{LW}^{x,u} \neq 0 \right) \right\}, \tag{13}$$

$$E_{S}^{X}(u) = \left\{ x \in X \ni \exists \left(\mathbf{S}^{x,u} \neq 0 \right) \right\}, \tag{14}$$

where

$$\mathbf{S}^{x,u}(t) \equiv \frac{\partial^e \mathbf{a}^u(t)}{\partial \mathbf{n}^x(t)^T} \tag{15}$$

is the sensitivity matrix.

Using these definitions, we can rewrite Eq. 12 as

$$\frac{\partial \mathbf{a}^u(t)}{\partial \mathbf{x}^T} = \frac{\partial^e \mathbf{a}^u(t)}{\partial \mathbf{x}^T} + \sum_{x \in E_S^X(u)} \mathbf{S}^{x,u}(t) \sum_{u' \in E_{LW}^U(x)} \sum_{d \in DL_{x,u'}} \frac{\partial^e \mathbf{n}^x(t)}{\partial \mathbf{a}^{u'}(t-d)^T} \frac{\partial \mathbf{a}^{u'}(t-d)}{\partial \mathbf{x}^T}. \tag{16}$$

The sensitivity matrix can be computed using static backpropagation, since it describes derivatives through a static portion of the network. The static backpropagation equation is

$$\mathbf{S}^{u,m}(t) = \left[\sum_{l \in L_b^m \cap E_S(u)} \mathbf{S}^{u,l}(t) \frac{\partial^e \mathbf{n}^l(t)}{\partial \mathbf{lz}^{l,m}(t,0)^T} \frac{\partial^e \mathbf{lz}^{l,m}(t)}{\partial \mathbf{a}^m(t)^T} \right] \dot{\mathbf{F}}^m\left(\mathbf{n}^m(t)\right), \; u \in U, \tag{17}$$

where m is decremented from u through the backpropagation order, L_b^m is the set of indices of layers that are directly connected backwards to layer m (or to which layer m connects forward) and that contain no delays in the connection, and

$$\dot{\mathbf{F}}(\mathbf{n}) = \begin{bmatrix} \frac{\partial f_1(\mathbf{n})}{\partial n_1} & \frac{\partial f_1(\mathbf{n})}{\partial n_2} & \cdots & \frac{\partial f_1(\mathbf{n})}{\partial n_S} \\ \frac{\partial f_2(\mathbf{n})}{\partial n_1} & \frac{\partial f_2(\mathbf{n})}{\partial n_2} & \cdots & \frac{\partial f_2(\mathbf{n})}{\partial n_S} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_S(\mathbf{n})}{\partial n_1} & \frac{\partial f_S(\mathbf{n})}{\partial n_2} & \cdots & \frac{\partial f_S(\mathbf{n})}{\partial n_S} \end{bmatrix}. \tag{18}$$

There are four terms in Eqs. 16 and 17 that need to be computed:

$$\frac{\partial^e \mathbf{n}^x(t)}{\partial \mathbf{a}^{u'}(t-d)^T}, \; \frac{\partial^e \mathbf{n}^l(t)}{\partial \mathbf{lz}^{l,m}(t,0)^T}, \; \frac{\partial^e \mathbf{lz}^{l,m}(t,0)}{\partial \mathbf{a}^m(t)^T}, \text{ and } \frac{\partial^e \mathbf{a}^u(t)}{\partial \mathbf{x}^T}. \tag{19}$$

The first term can be expanded as follows:

$$\frac{\partial^e \mathbf{n}^x(t)}{\partial \mathbf{a}^{u'}(t-d)^T} = \frac{\partial^e \mathbf{n}^x(t)}{\partial \mathbf{lz}^{l,m}(t,d)^T} \frac{\partial^e \mathbf{lz}^{l,m}(t,d)}{\partial \mathbf{a}^{u'}(t-d)^T} \tag{20}$$

The first term on the right of Eq. 20 is the derivative of the net input function, which is the identity matrix if the net input is the standard summation. The second term is the derivative of the weight function, which is the corresponding weight matrix if the weight function is the standard dot product. Therefore, the right side of Eq. 20 becomes simply a weight matrix for LDDN networks.

The second term in Eq. 19 is the same as the first term on the right of Eq. 20. It is the derivative of the net input function. The third term in Eq. 19 is the same as the second term on the right of Eq. 20. It is the derivative of the weight function.

The final term that we need to compute is the last term in Eq. 19, which is the explicit derivative of the network outputs with respect to the weights and biases in the network. One element of that matrix can be written

$$\frac{\partial^e a_k^u(t)}{\partial iw_{i,j}^{m,l}(d)} = \sum_{q=1}^{S} \frac{\partial^e a_k^u(t)}{\partial n_q^m(t)} \frac{\partial^e n_q^m(t)}{\partial iz_q^{m,l}(t,d)} \frac{\partial^e iz_q^{m,l}(t,d)}{\partial iw_{i,j}^{m,l}(d)} \tag{21}$$

The first term in this summation is an element of the sensitivity matrix, which is computed using Eq. 17. The second term is the derivative of the net input, and the third term is the derivative of the weight function. (We have made the assumption here that the net input function operates on each element individually.) Eq. 21 is the equation for an input weight. Layer weights and biases would have similar equations.

This completes the RTRL algorithm for networks that can be represented in the GLDDN framework. The main steps of the algorithm are Eqs. 11 and 16, where the components of Eq. 16 are computed using Eqs. 20 and 21. Computer code can be written from these equations, with modules for weight functions, net input functions and transfer functions added as needed. Each module should define the function response, as well as its derivative. The overall framework is independent of the particular form of these modules.

4 Conclusions

There are generally two different approaches for writing software to train dynamic networks. In the first approach, you use the general concepts of BPTT or RTRL to derive the dynamic backpropagation equations for a specific network architecture; each architecture has its own code. The second standard approach is to use some particular canonical form for dynamic networks. Such canonical forms can represent arbitrary dynamic networks. You can implement a dynamic backpropagation algorithm for the canonical form, and then you can transform the specific architecture in question into the canonical form in order to use the software.

In this paper we have taken a different approach. We have developed a framework that encompasses a very general class of network architectures, and then we have presented a set of equations that can be used to compute the gradients for any network that fits within that framework. This makes it possible to write software that is applicable to arbitrarily connected networks, without having to convert complex architectures into a standard canonical form.

Dynamic networks have applications in a wide variety of areas, and such disparate applications have lead to diverse network architectures. The ability to test new architectures quickly, without having to write problem-specific code or to convert each architecture into a canonical form, will enable the more rapid spread of dynamic neural networks into new fields.

References

1. M. Hagan, H. Demuth, O. De Jesús, "An Introduction to the Use of Neural Networks in Control Systems," invited paper, International Journal of Robust and Nonlinear Control, Vol. 12, No. 11 (2002) pp. 959-985.

2. J. Roman, and A. Jameel, "Backpropagation and recurrent neural networks in financial analysis of multiple stock market returns," Proceedings of the Twenty-Ninth Hawaii International Conference on System Sciences, vol. 2, (1996) pp. 454-460.
3. J. Feng, C. K. Tse, F. C. M. Lau, "A neural-network-based channel-equalization strategy for chaos-based communication systems," IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 50, no. 7 (2003) pp. 954-957.
4. I. Kamwa, R. Grondin, V. K. Sood, C. Gagnon, V. T. Nguyen, J. Mereb, "Recurrent neural networks for phasor detection and adaptive identification in power system control and protection," IEEE Transactions on Instrumentation and Measurement, vol. 45, no. 2, (1996) pp. 657-664.
5. Jayadeva and S. A. Rahman, "A neural network with O(N) neurons for ranking N numbers in O(1/N) time," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 51, no. 10, (2004) pp. 2044-2051.
6. G. Chengyu, and K. Danai, "Fault diagnosis of the IFAC Benchmark Problem with a model-based recurrent neural network," in Proceedings of the 1999 IEEE International Conference on Control Applications, vol. 2, (1999) pp. 1755-1760.
7. A. J. Robinson, "An application of recurrent nets to phone probability estimation," in IEEE Transactions on Neural Networks, vol. 5, no. 2 (1994).
8. L. R. Medsker, and L. C. Jain, Recurrent neural networks: design and applications, Boca Raton, FL: CRC Press (2000).
9. P. Gianluca, D. Przybylski, B. Rost, and P. Baldi, "Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles," in Proteins: Structure, Function, and Genetics, vol. 47, no. 2 , (2002) pp. 228-235.
10. P. J. Werbos, "Backpropagation through time: What it is and how to do it," Proceedings of the IEEE, vol. 78, (1990) pp. 1550–1560.
11. R. J. Williams, and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," Neural Computation, vol. 1, (1989) pp. 270–280.
12. O. De Jesús, and M. Hagan, "Backpropagation Algorithms for a Broad Class of Dynamic Networks," IEEE Transactions on Neural Networks, Vol. 18, No. 1 (2007) pp. 14 -27.
13. O. De Jesús, Training General Dynamic Neural Networks, Doctoral Dissertation, Oklahoma State University, Stillwater OK, (2002).
14. K. S. Narendra, and A. M. Parthasrathy, Identification and control for dynamic systems using neural networks, IEEE Transactions on Neural Networks, Vol. 1, No. 1 (1990) pp. 4-27.
15. E. Wan, and F. Beaufays, "Diagrammatic Methods for Deriving and Relating Temporal Neural Networks Algorithms," in Adaptive Processing of Sequences and Data Structures, Lecture Notes in Artificial Intelligence, Gori, M., and Giles, C.L., eds., Springer Verlag (1998).
16. G. Dreyfus, and Y. Idan, "The Canonical Form of Nonlinear Discrete-Time Models," Neural Computation 10, 133–164 (1998).
17. A. C. Tsoi, and A. Back, "Discrete time recurrent neural network architectures: A unifying review," Neurocomputing 15 (1997) 183-223.
18. L. Personnaz, and G. Dreyfus, "Comment on 'Discrete-time recurrent neural network architectures: A unifying review'," Neurocomputing 20 (1998) 325-331.
19. L. A. Feldkamp, and G. V. Puskorius, "A signal processing framework based on dynamic neural networks with application to problems in adaptation, filtering, and classification," Proceedings of the IEEE, vol. 86, no. 11 (1998) pp. 2259 - 2277.
20. P. Campolucci, A. Marchegiani, A. Uncini, and F. Piazza, "Signal-Flow-Graph Derivation of On-line Gradient Learning Algorithms," Proceedings of International Conference on Neural Networks ICNN'97 (1997) pp.1884-1889.

Neural Networks and Associative Memories

Infinite Dimensional Associative Memories: Foundations (with Theorems)

Enrique Carlos Segura
Department of Computer Science, University of Buenos Aires
Ciudad Universitaria, Pab.I, (1428) Buenos Aires, Argentina
e-mail: esegura@dc.uba.ar
(Paper received on July 06, 2007, accepted on September 1, 2007)

Abstract

We present a formal theoretical background, including theorems and their proofs, for our neural network model with associative memory and continuous topology, i.e. its processing units are elements of a continuous metric space and the state space is Euclidean and in¯nite dimensional. This approach is intended as a generalization of the previous ones due to Little and Hop¯eld. Thus we integrate two levels of continuity: continuous response units and continuous topology of the neural system, obtaining a more biologically plausible model of associative memory. A theoretical framework is provided so as to make this integration consistent. We ¯rst present some general results concerning attractors and stationary solutions. Then we focus on the case of orthogonal memories, proving theorems on their stability, size of attraction basins and spurious states. Finally, we get back to discrete models, i.e. we discuss new viewpoints arising from the present continuous approach and examine which of the new results are also valid for the discrete models.

Keywords: associative memory, continuous topology, dynamical systems, Hop¯eld model, in¯nite dimensional state space, stability.

1 Introduction

In seminal papers, Little [8],[9] and Hop¯eld [6] constructed a content-addressable memory as a dense network of arti¯cial neurons that are represented as elementary bistable processors. Addressability is guaranteed by the dissipative dynamics of the system. It consists of switching each processor from one of its stable con¯gurations to the other as a consequence of the intensity of the local ¯eld acting on it. The memories, corresponding to ¯xed points of the dynamics, are stored in the system in a distributed manner through the matrix of two-body interactions (synaptic e±cacies) between the neurons. If this matrix is properly de¯ned, the above dynamics is enough so as to ensure a monotonic decrease of an \energy" function. Thus, starting from an arbitrary con¯guration the system is led to a local minimum that corresponds to the nearest stored memory.

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 15-28

In a later paper, Hopfield [7] aimed at a more realistic model by replacing bistable neurons by graded response devices. In fact, a classical objection to the former model [6],[8],[9] was that a two-state representation of the neural output is, from a biological point of view, an oversimplification and that it is necessary to describe relevant neural activity by firing rates, rather than merely by the presence or the absence of an individual spike[1]. In either case the retrieval process is again guaranteed by the nature of the matrix of synaptic efficacies. However, in [7] the space of states describing the patterns of activity remained discrete, in the sense that the number of units was, at most, countable. This was an open gap in the plausibility of the model. In fact, since the Little model was formulated to describe the computational ability of an ensemble of simple processing units, it was necessary to reconcile the biological evidence of a true continuum of the neural tissue with the descriptions provided by discrete models inspired in an Ising system. While the empirical evidence always shows patterns of activity or quiescence involving patches with finite sizes, the ferromagnetic approach suggests systems with discrete processing units with no finite dimensions. In spite of this simplification all the discrete models have been remarkably successful in describing emergent processing abilities that correspond to stylized facts concerning basic elementary cognitive processes.

In this paper we introduce a solid theoretical background, including theorems and their proofs, for our neural network model with associative memory and processing units defined as elements of a continuous metric space (some subsidiary results are omitted by length limitations). This model [12],[13] is intended as a generalization of the previous ones due to Little and Hopfield. Our main purpose is to provide proofs in the sense that it is actually possible to formulate a system of associative memory (AM) with continuous response units and a continuous topological structure on the set of such units. We conceive the network so as to preserve the salient features that made attractive all the discrete models, especially the levels of continuity that the Hopfield model with graded response [7] added to the discrete one [6]: continuous-valued units and continuous scale of time, via the transition from discrete to continuous, differential equation dynamics. In spite of the fact that the corresponding space of configurations is an infinite dimensional functional space, we can define a basic simple dynamics having asymptotic, stationary solutions which can be associated to minima of an energy functional of Lyapunov type and can be taken to represent the memories stored in the system.

We have already introduced in a previous paper [12] several of the results included here, but without any rigorous proof. The present article is devoted to provide theoretical foundations for that sketch. We place emphasis on a detailed analysis of orthogonal memories, a relevant particular case of the general theory (deep comprehension of orthogonal memories is essential to understand the general pseudo-orthogonal case). However, we also present some other more general results.

Some approaches related to ours have appeared in recent years [10],[11]. The concept of *field computation*, introduced by MacLennan, has a similar inspiration since many neural phenomena can be described as a field, i.e. the distribution of some physical quantity over a continuous space, with a topology associated to it. On the other hand, the big number of neurons per sq. milimeter that can be found

[1]However, there is at present an increasing agreement that spiking neurons have some properties for describing certain aspects of neural dynamics not completely covered by rate-coding models.

throughout most of the brain cortex, justifies treatment of neural activity as a field.

All these arguments are related to our approach. However, this is aimed to a different purpose, which is that of formulating an extended model of AM and theoretically founding it, including justification of its potential as a tool for modelling cognitive processes of memory and learning.

Moreover, in another previous paper [13], we have already proposed a generalization of the nondeterministic, finite temperature Glauber dynamics [3] to the case of a finite number of graded response neurons (Hopfield'84). We did this by casting the retrieval process of a Hopfield model with continuous-valued units, into the framework of a diffusive process governed by the Fokker-Plank equation. We thus provided a description of the transitional regime that rules the retrieval process, which is currently disregarded. In other words, we unified the graded response units model [7] and the stochastic approach, obtaining a description of the retrieval process at both the microscopic, individual neuron level and the macroscopic level of time evolution of the probability density function over the space of activation patterns, i.e. an equation describing, for each possible pattern, how, given an initial probability for the system being in it, this probability changes upon time.

The paper is organized as follows. In Sect. 2 we give basic concepts and definitions. Sect. 3 provides general results on attractors and stationary solutions. In Sect. 4, we focus on the orthogonal case, proving theorems on stability of the memories and of the origin. Sect. 5 presents a result on the size of basins of attraction and Sect. 6 deals with spurious states. Finally, in Sect. 7 we get back to discrete models, i.e. discuss new viewpoints arising from the present continuous approach and examine which of the new results are also valid for the discrete models.

2 Basic Definitions and First Results

Assume that $\mathbf{v}(x,t)$ describes the activity of a point-like neuron located in x at time t. This pattern of activity evolves according to:

$$\frac{\partial \mathbf{v}(x,t)}{\partial t} = -\mathbf{v}(x,t) + g_\sigma \left(\int_K \mathbf{T}(x,y)\mathbf{v}(y,t)dy \right) \tag{1}$$

with $\mathbf{v}(x,t) : K \times R_{\geq 0} \rightarrow R$, $K \subset X$. X is a metric space, K a compact domain, g_σ a *sigmoid* function, i.e. $g_\sigma \in C^1(R)$, non decreasing, odd and satisfying $\lim_{x \rightarrow \pm\infty} g_\sigma(x) = \pm \mathrm{V_M}$, $\lim_{\sigma \rightarrow \infty} g_\sigma(x) = sgn(x)\ \forall x \neq 0$, $\mid g_\sigma(x) \mid < \min\{\mathrm{V_M}, \sigma x\}$ and $g'_\sigma(0) = \sigma$.

Let S be the set of all possible states $\mathbf{v}(x)$ (patterns of activity) of the system. Then a solution $\mathbf{v}(x,t)$ fulfilling (1) is a trajectory in S.

We can assume that $\mathrm{V_M} = 1$. As for $\mathbf{T}$:$K \times K \rightarrow R$, we assume it is continuous almost everywhere (a.e.) in order to ensure that the integral is well defined. As a natural extension of the discrete case we introduce the *local field* on (or net input to) the neuron located in x when the state of the system is $\mathbf{v}(y,t)$:

$$h_t^{\mathbf{v}}(x) = \int_K \mathbf{T}(x,y)\mathbf{v}(y,t)dy$$

For $t = 0$ we write $h^{\mathbf{v}}(x) = h_0^{\mathbf{v}}(x)$. Note that $h^{\mathbf{v}}$ is linear in $\mathbf{v}$.

Let $\mathbf{v}_0^{\mu}(x) = \mathbf{v}^{\mu}(x, 0)$ be an initial condition (i.c.) and $\mathbf{v}(x, t)$ the solution of (1) associated to it. We say that $\mathbf{v}^{\mu}(x)$ is a *memory* or an *attractor* if and only if:

1) $\mathbf{v}^{\mu}$ is an equilibrium point, i.e. $\mathbf{v}^{\mu}(x) = g_{\sigma}(h_t^{\mathbf{v}^{\mu}}(x))$ *a.e.*

2) For every $t_0 \geq 0$ and $\mathbf{v}_0$ a different i.c. corresponding to $\mathbf{v}$, there exists $\delta(t_0) > 0$ such that if $\| \mathbf{v}^{\mu} - \mathbf{v}_0 \| < \delta$ then $\| \mathbf{v}^{\mu}(\cdot, t) - \mathbf{v}(\cdot, t) \| \rightarrow 0$ when $t \rightarrow \infty$.

Hence, attractors are stationary solutions of (1). Assume that $\mathbf{S}=L^2(K)$ and that $| K | < \infty$ (K has finite Lebesgue measure).

We define the *energy* of the system at time t_0 as the functional:

$$\mathbf{H}[\mathbf{v}(\cdot, t_0)] = -\frac{1}{2} \int_K \int_K \mathbf{T}(x, y)\mathbf{v}(x, t_0)\mathbf{v}(y, t_0)dxdy + \int_K \int_0^{\mathbf{v}(x,t_0)} g_{\sigma}^{-1}(s)dsdx \qquad (2)$$

where $\mathbf{H}[\mathbf{v}(\cdot, t_0)]$ means that $\mathbf{v}$ is viewed as a function of x. Thus, each $\mathbf{v}$ in $\mathbf{S}$ has an energy $\mathbf{H}(\mathbf{v})$. This is an extension of the energy as defined in [7] for the (discrete) model with graded response functions.

2.1 Attractors and Stationary Solutions

From now on we assume that $\mathbf{T}$ is symmetric.

Theorem 2.1: $\mathbf{H}$ is monotonically decreasing with t and reaches its minima at $\mathbf{v}_{t_e}(x) = \mathbf{v}(x, t_e)$ such that

$$\left[\frac{\partial \mathbf{v}}{\partial t}(x, t)\right]_{t_e} = 0 \qquad (3)$$

a.e. in K (in words, given a solution $\mathbf{v}(x, t)$ corresponding to some i.c., the minima of $\mathbf{H}$ are equilibrium points of the system). This theorem generalizes the classical result for the discrete Hopfield model with graded responses [7] (see e.g. [1],[5]).

Proof: omitted by length limitation.

Memories or attractor states, as defined in this section, satisfy the above conditions. However, the reciprocal implication is not necessarily true: from the previous theorem it does not follow that if a solution $\mathbf{v}(x, t)$ of (1) satisfies condition (3) for some t^*, then $\mathbf{v}(x, t^*)$ is an attractor. For example, the trivial solution $\mathbf{v} \equiv 0$ satisfies it for all t but, as we soon will see, its stability or instability depends on the slope σ of g_{σ} at the origin. In general, the possibility to construct nontrivial memories strongly depends on such parameter.

The sigmoid function g_{σ} plays an important role in determining in which cases the system has nontrivial stationary solutions. A necessary condition is given by:

Theorem 2.2: (existence and uniqueness of the solution) If $\sigma < \frac{1}{\mathrm{M}|K|}$, being M such that $| \mathbf{T}(x, y) | \leq \mathrm{M}$, then the unique stationary solution of (1) is $\mathbf{v} \equiv 0$.

Proof: by definition of g_σ, σ is a Lipschitz constant for it. Then, assuming that $\mathbf{v}^1$ and $\mathbf{v}^2$ are two fixed points of the operator A defined as

$$A\mathbf{v} = g_\sigma \left[\int_K \mathbf{T}(x,y)\mathbf{v}(y)dy \right]$$

we get (using the L^2 norm):

$$\mid A\mathbf{v}^1(x) - A\mathbf{v}^2(x) \mid = \mid g_\sigma(\int_K \mathbf{T}(x,y)\mathbf{v}^1(y)dy) - g_\sigma(\int_K \mathbf{T}(x,y)\mathbf{v}^2(y)dy) \mid$$

$$\leq \sigma \mid \int_K \mathbf{T}(x,y)\mathbf{v}^1(y)dy - \int_K \mathbf{T}(x,y)\mathbf{v}^2(y)dy \mid \leq \sigma M \mid K \mid^{\frac{1}{2}} \parallel \mathbf{v}^1 - \mathbf{v}^2 \parallel$$

being M an upper bound for $\mid \mathbf{T}(x,y) \mid$ (which exists since $\mathbf{T}$ is continuous and K is compact). Finally:

$$\parallel A\mathbf{v}^1 - A\mathbf{v}^2 \parallel \leq \mid K \mid^{\frac{1}{2}} \sup_{x \in K} \mid A\mathbf{v}^1(x) - A\mathbf{v}^2(x) \mid < \sigma M \mid K \mid \parallel \mathbf{v}^1 - \mathbf{v}^2 \parallel$$

Then A is a contraction and has a unique fixed point provided $\sigma < \frac{1}{M \mid K \mid}$.

Besides the condition $\sigma \mathrm{M} \mid K \mid \geq 1$, other ones (see next section) have to be fulfilled in order to ensure the actual existence of nontrivial solutions.

3 Orthogonal Memories, Hebbian Synapses

The case we are specially interested in is the storage of orthogonal memories when the matrix of synaptic weights is Hebbian. This can be achieved through a straightforward generalization of the Hebb rule [4]. Let $\{\mathbf{v}^\mu\}$ be an orthogonal set of functions in some space $S(K)$, that is to say $(\mathbf{v}^\mu, \mathbf{v}^\nu) = 0$ if $\mu \neq \nu$. In principle, $S(K)$ may be noncountable and hence we can define in general:

$$\mathbf{T}(x,y) = \frac{1}{\mid K \mid} \int_P \mathbf{v}^\rho(x)\mathbf{v}^\rho(y)d\rho$$

for $\rho \in P$ some index set. In the case $\{\mathbf{v}^\mu\}$ is an orthogonal set in $L^2(K)$, it is at most countable (provided the separability of $L^2(K)$). Therefore it is natural to restrict to the case in which P is countable:

$$\mathbf{T}(x,y) = \frac{1}{\mid K \mid} \sum_{\mu=1}^{p} \mathbf{v}^\mu(x)\mathbf{v}^\mu(y) \tag{4}$$

Then the following theorem holds:

Theorem 3.1: The system (1), with $\mathbf{T}(x,y)$ defined as in (4), may have any finite number p of orthogonal memories taking values in $\{\mathrm{V}_*, -\mathrm{V}_*\}$, with $g_\sigma(\pm \mathrm{V}_*^3) = \pm \mathrm{V}_*$.

Proof: let p be a positive integer and $\{\mathbf{v}^\mu\}_{\mu=1}^{p} \subset L^2(K)$, $(\mathbf{v}^\mu, \mathbf{v}^\nu) = 0$ if $\mu \neq \nu$, $\mathbf{v}^\mu(x) \in \{-V_*, V_*\}$, $1 \leq \mu \leq p$, $x \in K$, with V_* such that $g_\sigma(\pm V_*^3) = \pm V_*$ (which exists and depends on σ). Defining $\mathbf{T}$ by (4) it holds that, for any μ:

$$g_\sigma(h^\mu(x)) = g_\sigma(\int_K \mathbf{T}(x,y)\mathbf{v}^\mu(y)dy) = g_\sigma(\int_K \frac{1}{|K|}\sum_{\nu=1}^{p} \mathbf{v}^\nu(x)\mathbf{v}^\nu(y)\mathbf{v}^\mu(y)dy)$$

Since the integrals are finite, we can interchange the sum and the integration:

$$g_\sigma(\frac{1}{|K|}\sum_{\nu=1}^{p} \mathbf{v}^\nu(x)\delta_{\nu\mu} \parallel \mathbf{v}^\mu \parallel^2) = g_\sigma(\frac{\mathbf{v}^\mu(x)\, V_*^2\, |K|}{|K|}) = \mathbf{v}^\mu(x)$$

Then, $\mathbf{v}^\mu(x,t) = \mathbf{v}^\mu(x)$ $\forall\, t > 0$ is a fixed point of equation (1).

These solutions $\mathbf{v}^\mu(x)$ look like the example depicted in Figure 1. Activation patterns of this kind agree with the intuitive generalization of the attractors of an Ising-type, spin glass discrete neural network in which patches of full activation alternate randomly with those of full quiescence. They can also be viewed as the vertices of an infinite (noncountable) dimensional hypercube.

A question arising is whether the set of orthogonal fixed points of (1) can be infinite. Note first that it is countable: the elements $\mathbf{v}^\mu$ as they were defined belong to $L^2(K)$, a separable space; hence every orthogonal set in it must be countable. However even an infinite countable number of orthogonal fixed points is not possible while preserving the integrability of $\mathbf{T}$. Observe that if there are k_μ changes of sign in $\mathbf{v}^\mu$ then each term of the form $\mathbf{v}^\mu(x)\mathbf{v}^\mu(y)$ divides the domain $K \times K$ in $(k_\mu + 1)^2$ square regions. Moreover, each region is separated from the next by discontinuity lines because such term takes the constant values $+\mathrm{V}_*^2$ or $-\mathrm{V}_*^2$. If the set of memories is infinite, the number of terms in $\mathbf{T}$ that are added is also infinite, therefore those discontinuity lines are dense at least in a neighborhood of some point, and $\mathbf{T}$ ceases to be piecewise continuous.

Note that theorem 3.1 implies a qualitative difference between discrete and continuous models. Since the memory capacity is now unbounded, there is nothing like a "phase diagram" in which, for a domain K and above some critical number p_c of memories, a transition to a "confusion phase" takes place, implying a rapid degradation of the retrieval ability. While in discrete models the size of the domain is determined by the dimension of the state space ($p_c = \alpha_c N$), in the present case this dimension is infinite and hence there is no p_c.

Discussions on discrete models are mostly in the thermodynamic limit in which either the number of neurons and the number of memories tend to infinity while their ratio is kept constant. This is not possible in the continuous limit, but it is certainly not a problem as far as the biological plausibility of the model is concerned.

We end this section with the following results that are easy to check.

Lemma 3.2: If the memories are orthogonal, the distance between any two of them is always the same.

Corollary 3.3: Orthogonal memories are never dense in $L^2(K)$.

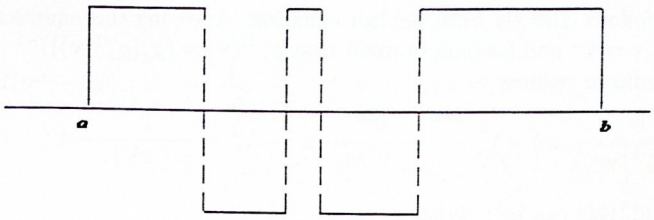


Figure 1: A memory in the space S=$L^2[a,b]$.

4 Stability of the Solutions

We will now derive conditions for the elements $\mathbf{v}^\mu$, as defined in Section 3, to be stable equilibria (i.e. memories) for equation (1).

Theorem 4.1: elements $\mathbf{v}^\mu$ are stable fixed points of (1) if and only if $g'_\sigma(\mathrm{V}_*^3) < \frac{1}{\mathrm{V}_*^2}$.

Theorem 4.2: (Stability of the origin) The solution $\mathbf{v} \equiv 0$ is stable if and only if $g'_\sigma(0) = \sigma < \frac{1}{\mathrm{V}_*^2}$.

Proof (both theorems): the computation of the directional derivatives of $\mathbf{H}(\mathbf{v})$ at an arbitrary point yields:

$$D_w\mathbf{H}(\mathbf{v}) = -\frac{1}{\mid K \mid}\sum_{\nu=1}^{p}(\mathbf{v}^\nu,\mathbf{v})(\mathbf{v}^\nu,w) + (g_\sigma^{-1}(\mathbf{v}),w)$$

with $w \in L^2(K)$ and $\parallel w \parallel = 1$. Now, if $\mathbf{v} = \mathbf{v}^\mu$, using the condition of orthogonality and noting that $\parallel \mathbf{v}^\mu \parallel^2 = \mathrm{V}_*^2 \mid K \mid$, it follows that $D_w\mathbf{H}(\mathbf{v}) = 0$. Similarly, it is easy to check that $D_w\mathbf{H}(\mathbf{v})$ vanishes for any element in $span\,\{\mathbf{v}^\mu\}_{\mu=1}^p$, i.e. linear combinations of the memories, when those combinations take values on $\{\mathrm{V}_*, -\mathrm{V}_*, 0\}$.

$$D_{w^2}^2\mathbf{H}(\mathbf{v}) \;=\; -\frac{1}{\mid K \mid}\sum_{\nu=1}^{p}(\mathbf{v}^\nu,w)^2 + (\frac{\partial}{\partial\mathbf{v}}g_\sigma^{-1}(\mathbf{v})w,w)$$

But the $\mathbf{v}^\nu$ are assumed orthogonal. Therefore, the use of Bessel's inequality yields:

$$\sum_{\nu=1}^{p}\frac{(\mathbf{v}^\nu,w)^2}{\parallel \mathbf{v}^\nu \parallel^2} \leq \parallel w \parallel^2 = 1 \Longleftrightarrow \sum_{\nu=1}^{p}(\mathbf{v}^\nu,w)^2 \leq \mathrm{V}_*^2 \mid K \mid$$

hence

$$D_{w^2}^2\mathbf{H}(\mathbf{v}) \geq (\frac{\partial}{\partial\mathbf{v}}g_\sigma^{-1}(\mathbf{v})w,w) - \mathrm{V}_*^2$$

for any w in S, $\parallel w \parallel = 1$. Then, a necessary and sufficient condition for an element $\mathbf{v}$ in S to be a minimum of $\mathbf{H}$ is:

$$(g_\sigma^{-1\prime}(\mathbf{v})w,w) - \mathrm{V}_*^2 > 0 \;\; \forall\, w \in \mathrm{S}, \parallel w \parallel = 1$$

Theorem 4.2 follows directly from the last equation. Applying this equation to the case in which $\mathbf{v} = \mathbf{v}^\mu$ and keeping in mind that $g_\sigma^{-1\prime}(\mathbf{v}) = (g'_\sigma(g_\sigma^{-1}(\mathbf{v})))^{-1}$ the above condition reduces to

$$(\frac{w}{g'_\sigma(g_\sigma^{-1}(\mathbf{v}^\mu))}, w) - \mathrm{V}_*^2 = (\frac{w}{g'_\sigma(\mathrm{V}_*^2\mathbf{v}^\mu)}, w) - \mathrm{V}_*^2 = \frac{1}{g'_\sigma(\pm\mathrm{V}_*^3)} - \mathrm{V}_*^2 > 0$$

Since g_σ is odd, this can be rewritten as:

$$g'_\sigma(\mathrm{V}_*^3) < \frac{1}{\mathrm{V}_*^2} \text{ or } g'_\sigma(\mathrm{V}_*^3)\mathrm{V}_*^2 < 1$$

Let us compare the necessary and sufficient condition given by theorem 4.2 for the stability of the origin with the uniqueness condition for the general case (theorem 2.2). When $\mathbf{T}$ is defined according to (4), the $\mathbf{v}^\mu$'s being stationary solutions of (1) and therefore $\mathbf{v}^\mu(x) \in \{\mathrm{V}_*, 0, -\mathrm{V}_*\}$, we have:

$$\mid \mathbf{T}(x,y) \mid \leq \frac{p\mathrm{V}_*^2}{\mid K \mid} = \mathrm{M}.$$

In this case the condition for the origin to be the only solution is that $\sigma < \frac{1}{\mathrm{M}\mid K\mid} = \frac{1}{p\mathrm{V}_*^2}$. This is more restrictive than what follows from theorem 4.2. Therefore, for the case of orthogonal memories there exists an intermediate range for the values of σ ($\sigma \in [\frac{1}{p\mathrm{V}_*^2}, \frac{1}{\mathrm{V}_*^2}]$, which degenerates into a point if $p = 1$) for which the trivial solution $\mathbf{v} \equiv 0$ is an attractor, but not necessarily the only solution of (1). Note, in addition, that the conditions derived in theorems 4.1 y 4.2 are independent of p (number of memories); this is a consequence of the orthogonality of the memories.

5 Basins of Attraction

Using the preceeding results, we can now estimate the size of the basins of attraction.

Theorem 5.1: for $p \geq 2$, the largest sphere contained in the basin of attraction of an orthogonal memory $\mathbf{v}^\mu$, $1 \leq \mu \leq p$, has a radius $k = \mathrm{V}_* \sqrt{\frac{\mid K \mid}{2}}$.

In other words, whenever $\| \mathbf{v}^\mu - \mathbf{v}_0 \| < k$, the distance $\| \mathbf{v}^\mu(\cdot,t) - \mathbf{v}(\cdot,t) \| \to 0$ when $t \to \infty$ (being $\mathbf{v}_0$ any i.c. for (1) and $\mathbf{v}$ the corresponding solution).

Proof: the radius of the basin will be the largest number $k > 0$ such that $D_w\mathbf{H}(\mathbf{v}^\mu + kw) > 0 \quad \forall w \in S, \| w \| = 1$. We know that

$$D_w\mathbf{H}(\mathbf{v}) = -\frac{1}{\mid K \mid}\sum_{\nu=1}^{p}(\mathbf{v}^\nu, \mathbf{v})(\mathbf{v}^\nu, w) + (g_\sigma^{-1}(\mathbf{v}), w)$$

Then:

$$\begin{aligned} D_w \mathbf{H}(\mathbf{v}^\mu + kw) &= -\frac{1}{\mid K \mid} \sum_{\nu=1}^{p} (\mathbf{v}^\nu, \mathbf{v}^\mu + kw)(\mathbf{v}^\nu, w) + (g_\sigma^{-1}(\mathbf{v}^\mu + kw), w) \\ &= \frac{1}{\mid K \mid} \left\{ V_*^2 \mid K \mid (\mathbf{v}^\mu, w) + k \sum_{\nu=1}^{p} (\mathbf{v}^\nu, w)^2 \right\} \\ &\quad + (g_\sigma^{-1}(\mathbf{v}^\mu + kw), w) \end{aligned}$$

which is positive if and only if

$$(g_\sigma^{-1}(\mathbf{v}^\mu + kw), w) > \frac{1}{\mid K \mid} \left\{ V_*^2 \mid K \mid (\mathbf{v}^\mu, w) + k \sum_{\nu=1}^{p} (\mathbf{v}^\nu, w)^2 \right\}$$

for every direction w. By virtue of Bessel's inequality:

$$\sum_{\nu=1}^{p} \frac{(\mathbf{v}^\nu, w)^2}{\| \mathbf{v}^\nu \|^2} \leq \| w \|^2 = 1$$

and, consequently (remembering that $\| \mathbf{v}^\nu \|^2 = V_*^2 \mid K \mid$), the condition is satisfied by imposing $(g_\sigma^{-1}(\mathbf{v}^\mu + kw), w) > (g_\sigma^{-1}(\mathbf{v}^\mu), w) + kV_*^2$ or equivalently

$$\int_K \frac{g_\sigma^{-1}(\mathbf{v}^\mu(x) + kw) - (g_\sigma^{-1}(\mathbf{v}^\mu(x))}{k} w(x)\, dx > V_*^2 \tag{5}$$

In order to prove that inequality (5) holds no matter the direction w, let us take the worst case: w pointing to a different memory, say $\mathbf{v}^\nu$, i.e. $w = \frac{\mathbf{v}^\nu - \mathbf{v}^\mu}{\|\mathbf{v}^\nu - \mathbf{v}^\mu\|}$. It is easy to check that $\mathbf{v}^\nu - \mathbf{v}^\mu$ can take only values 0 and $\pm 2V_*$ and that, by virtue of the orthogonality, it is 0 exactly on one half of the domain K and $\pm 2V_*$ on the other half. Then w is 0 on a subdomain of size $\frac{|K|}{2}$ and $\sqrt{\frac{2}{|K|}}$ on the remaining subdomain of equal size. Thus, condition (5) can be rewritten as

$$\frac{\mid K \mid}{2} \frac{g_\sigma^{-1}(\mathbf{v}^\mu(x) + k\sqrt{\frac{2}{|K|}}) - (g_\sigma^{-1}(\mathbf{v}^\mu(x))}{k\sqrt{\frac{2}{|K|}}} \frac{2}{\mid K \mid} > V_*^2$$

(multiplying numerator and denominator by $\sqrt{\frac{2}{|K|}}$), which holds if

$$k\sqrt{\frac{2}{\mid K \mid}} < V_* \quad \Longleftrightarrow \quad k < V_* \sqrt{\frac{\mid K \mid}{2}}$$

Finally, the largest spherical basin of attraction has a radius equal to $V_* \sqrt{\frac{|K|}{2}}$, since otherwise the basins would not be disjoint, because the distance between any two $\mathbf{v}^\mu$ and $\mathbf{v}^\nu$ is twice that quantity.

Note that this result does not imply that the basins of attraction are spherical. It only limits the radius of *spherical* basins for memories $\mathbf{v}^\mu$ and, consequently, for $-\mathbf{v}^\mu$ as well. Figure 2 shows a simplified bidimensional sketch.

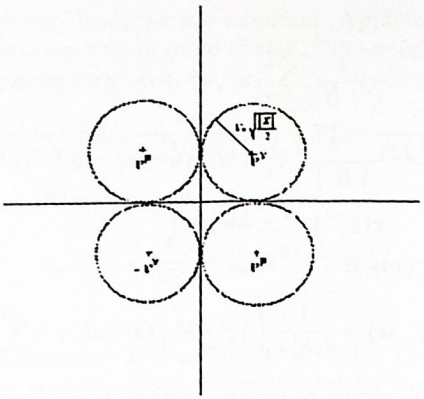


Figure 2: Two orthogonal memories and their inverses, each one with norm $V_* \sqrt{|K|}$ and a spherical basin of attraction of radius $V_* \sqrt{\frac{|K|}{2}}$.

6 Spurious Memories

As a consequence of the nonlinearity of the dynamics under consideration, undesired fixed points appear in addition to those purposely stored in the synaptic operator $\mathbf{T}$ with the Hebb prescription. These are called *spurious states* or *spurious memories*.

It is possible to distinguish two types of spurious states: *mixture* and *non-mixture* memories. $\mathbf{v}$ is said to be a mixture state if it can be expressed as a linear combination of the stored memories: $\mathbf{v} = \sum_{i=1}^{q} \alpha_{\mu_i} \mathbf{v}^{\mu_i}$ with $q \leq p$, $\mathbf{v}^{\mu_i} \in \{\mathbf{v}^{\mu}\}$ and α_{μ_i} real constants. If no such $\{\alpha_{\mu}\}$ exists, the spurious state is said non-mixture.

6.1 Mixture Spurious States

First note that, just like in the known discrete models, for every memory $\mathbf{v}^{\mu}$, $-\mathbf{v}^{\mu}$ is also a memory. In the simple case when $p = 1$, there exist only two spurious states: the origin ($\mathbf{v} \equiv 0$) and the inverse of the (unique) stored memory. Thus, there are no non-mixture states for $p = 1$. If $p \geq 2$, the analysis gets considerably harder.

We have already mentioned the fact that every mixture state is a fixed point if $\mathbf{v}(x) \in \{V_*, -V_*, 0\}$ $\forall x \in \omega$. This can be easily seen either by using the linearity of $h^{\mathbf{v}}$ or from the proof of theorems 4.1 and 4.2. It is also clear that only a small subset of $span\,\{\mathbf{v}^{\mu}\}_{\mu=1}^{p}$ contains spurious states. In particular, it follows that if $\mathbf{v}^{\mu}$ and $\mathbf{v}^{\nu}$ are memories, then $\pm\frac{1}{2}\mathbf{v}^{\mu} \pm \frac{1}{2}\mathbf{v}^{\nu}$ are fixed points of the dynamics. This implies that there exist at least $4\binom{p}{2}$ spurious (mixture) states. These are in general unstable, as stated by the following

Theorem 6.1: If $\mathbf{v}$ is a mixture spurious memory and there exists $x \in K$ such that $\mathbf{v}(x) = 0$, then $\mathbf{v}$ is a saddle point of the dynamics.

Proof: let $\mathbf{v} = \sum_{\mu=1}^{q} \alpha_\mu \mathbf{v}^\mu$, $1 \leq q \leq p$ (renaming memories if necessary). $\mathbf{v}$ is piecewise constant (since so are the $\mathbf{v}^\mu$'s, and there is a finite number of them). Therefore, if $\mathbf{v}(x) = 0$ then it vanishes in a neighborhood U of x and it holds that $0 < \sum_{i=1}^{I} \alpha_{\mu_i} \mathbf{v}^{\mu_i} = -\sum_{j=1}^{J} \alpha_{\mu_j} \mathbf{v}^{\mu_j}$ at every point in U, being $\{\mathbf{v}^{\mu_i}\}_{i=1}^{I} \cup \{\mathbf{v}^{\mu_i}\}_{j=1}^{J} = \{\mathbf{v}^{\mu_i}\}_{\mu=1}^{q}$

Let us choose $w = \sum_{i=1}^{I} \alpha_{\mu_i} \mathbf{v}^{\mu_i} - \sum_{j=1}^{J} \alpha_{\mu_j} \mathbf{v}^{\mu_j}$ (we can neglect the normalizing constant).

Then $w = 2 \sum_{i=1}^{I} \alpha_{\mu_i} \mathbf{v}^{\mu_i}$ in U. Now compute

$$g_\sigma(h^{\mathbf{v}+\varepsilon w}(x)) = g_\sigma(h^{\mathbf{v}}(x) + \varepsilon h^w(x))$$

Keeping in mind that $h^{\mathbf{v}}(x) = V_*^2 \mathbf{v}(x)$ (by hypothesis and by virtue of the linearity of $h^{\mathbf{v}}$) and computing

$$h^w(x) = \sum_{\mu=1}^{q} \alpha_\mu h^{\mathbf{v}^\mu}(x) = V_*^2 \sum_{\mu=1}^{q} \alpha_\mu \mathbf{v}^\mu(x) = 2V_*^2 \sum_{i=1}^{I} \alpha_{\mu_i} \mathbf{v}^{\mu_i}(x)$$

(which holds for every $x \in U$), we obtain

$$g_\sigma(h^{\mathbf{v}+\varepsilon w}(x)) = g_\sigma(V_*^2 \mathbf{v}(x) + 2\varepsilon V_*^2 \sum_{i=1}^{I} \alpha_{\mu_i} \mathbf{v}^{\mu_i}(x))$$

in U. The dynamics at $\mathbf{v} + \varepsilon w$ is (always in $U \subset K$):

$$\frac{\partial(\mathbf{v} + \varepsilon w)}{\partial t} = -2\varepsilon \sum_{i=1}^{I} \alpha_{\mu_i} \mathbf{v}^{\mu_i} + g_\sigma(2\varepsilon V_*^2 \sum_{i=1}^{I} \alpha_{\mu_i} \mathbf{v}^{\mu_i})$$

No matter the sign of $\sum_{i=1}^{I} \alpha_{\mu_i} \mathbf{v}^{\mu_i}$, the stability condition at $\mathbf{v}+\varepsilon w$ is $g_\sigma(2\varepsilon V_*^2 \sum_{i=1}^{I} \alpha_{\mu_i} \mathbf{v}^{\mu_i}) < 2\varepsilon \sum_{i=1}^{I} \alpha_{\mu_i} \mathbf{v}^{\mu_i}$ which is equivalent to $\sigma = g'_\sigma(0) < \frac{1}{V_*^2}$ which is false since, by hypothesis, $g_\sigma(\pm V_*^3) = g_{\sigma V_*^2}(\pm V_*) = \pm V_*$ (otherwise, g_σ would not have any fixed point apart from the origin). Then $\mathbf{v}$ is unstable in the direction w.

Now let us choose $w = \mathbf{v}$. With a similar reasoning, we get $g_\sigma(h^{\mathbf{v}+\varepsilon w}(x)) = g_\sigma(V_*^2 \mathbf{v}(1+\varepsilon))$ and the stability condition at $\mathbf{v} + \varepsilon w$ is $g_\sigma(V_*^2 \mathbf{v}(1+\varepsilon)) < \mathbf{v}(1+\varepsilon)$. Remembering that $\mathbf{v} = g_\sigma(V_*^2 \mathbf{v})$, the condition for the system to be stable at $\mathbf{v}+\varepsilon w$ is $g'_\sigma(V_*^3) < \frac{1}{V_*^2}$, which holds since the memories $\mathbf{v}^\mu$ are minima of $\mathbf{H}$ (cf. Sect. 2).

This leads to the useful

Corollary 6.2: The basins of attraction for mixture states have zero radius, in the sense of the $L^2(K)$ norm.

Example: for $q = 2$, all combinations of the form $\mathbf{v} = \pm\frac{1}{2}\mathbf{v}^\mu \pm \frac{1}{2}\mathbf{v}^\nu$ are spurious states (see Fig. 3). The direction of maximum unstability is given by $\mathbf{v}^\mu - \mathbf{v}^\nu$ and that of maximum stability is $\pm\mathbf{v}$ (directly towards or from the origin of coordinates).

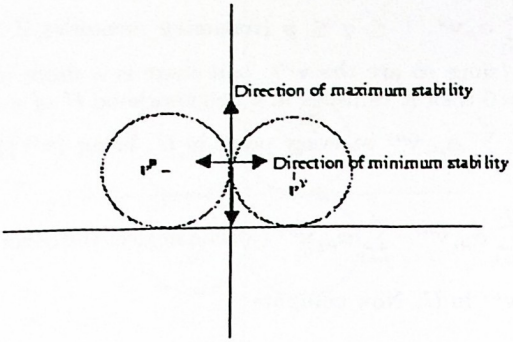


Figure 3: A mixture spurious state. Dotted lines indicate limits for the spherical basins of the two memories which compose the spurious state.

6.2 Non-mixture Spurious States

Unlike mixture spurious states, which can be calculated analytically, the non-mixture ones are difficult to find. Indeed, in the limit $p \to \infty$, the following property holds, no matter how $\mathbf{T}$ is defined.

Lemma 6.3: if $\{\mathbf{v}^\mu\}_{\mu=1}^{\infty}$ is complete[2] and $p \to \infty$, there are no non-mixture spurious memories.

Remark: since S=$L^2(K)$ here, a question about the meaning of lemma 6.3 may arise, i.e. is there some basis of $L^2(K)$ whose elements take on only two values, say $\pm V_*$? The answer is yes. The relevant example for $K \subset R$ (that can be extended to R^n) are the *Haar wavelets*, which form an orthonormal and complete basis in $L^2(-\infty, +\infty)$. They are bi-valued, but since they are normalized, such values change from one function to another. If normality is relaxed, it is possible to force them to take values in $\{V_*, -V_*\}$. If restricted to a bounded interval $K \subset R$, they form an orthogonal (but not orthonormal) and complete set in $L^2(K)$. However, if we construct $\mathbf{T}$ according to (4), this completeness can be only asymptotical: as we saw, the number of orthogonal memories can be as large as desired, but not infinite.

7 Back to the Discrete Domain

In order for the results of Sect. 2 to be valid, the only condition on X is to be a metric space (continuous or discrete). Therefore, all theorems of Sect. 2 hold for the discrete Hopfield model with continuous activities [7], if we replace the $L^2(K)$ norm with the usual euclidean norm and $\mid K \mid$ with N (number of neurons).

Concerning the results of Section 3, the situation is different. Clearly, theorem 3.1 is no longer true and the same happens for lemmas and corollaries based on

[2]The set $\{\mathbf{v}^\mu\}$ is said to be *complete* in $L^2(K)$ if and only if the minimal subspace of $L^2(K)$ which contains it is the entire space $L^2(K)$.

the possibility of memories with an unbounded number of discontinuities, such as corollary 3.3 (of no meaning for the discrete case). Instead, results concerning stability (Sect. 4) and size of attraction basins (Sect. 5) remain valid, with slight changes. The same is true for Section 6 (spurious states), except for theorem 6.3 (non-mixture spurious memories), which has no meaning for the discrete model.

As for the Hopfield model with discrete activities [6], first remark that its metrics based on the Hamming distance is not euclidean. However, being this metrics the discrete version of the L^1 norm, which is equivalent[3] to the quadratic L^2 norm, some results remain qualitatively true, e.g. theorems 2.1 and 2.3; but the mathematical tools used here are of no help to obtain them. And, on the other hand, the results of Sections 3 to 6 have no meaning in general (when based on concepts of euclidean distance and directional derivatives, of no application in the discrete case).

8 Conclusions

We introduced a formal theoretical background, including theorems and their proofs, for our neural network model with AM in which processing units are elements of a continuous metric space. This approach is intended as a generalization of the previous ones due to Little and Hopfield. Our main purpose was to provide a mathematical foundation of the actual possibility to formulate a system that unifies graded response units and continuous topological structure on the set of such units, obtaining a more biologically plausible model of AM.

On the other hand, our approach preserves salient features that made attractive all discrete models, especially the levels of continuity that the second Hopfield model [7] added to the discrete one [6]: graded activation functions and continuous scale of time, via the transition from discrete to continuous, differential equation dynamics.

Firstly (Section 2) general results were proved assuming only a symmetric weight matrix $\mathbf{T}$ with non-negative diagonal elements. These results are generalizations of well known properties of discrete, Ising-type models.

Then (Sections 3 to 6) we analyzed the case of orthogonal memories and a synaptic operator constructed through the autocorrelation (Hebb) rule. We proved:
-*Hebb rule*: it can be naturally extended to the infinite dimensional case.
-*Capacity*: any finite set of orthogonal memories can be stored and retrieved. However there are some differences in capacity with regard to discrete approaches.
-*Stability*: necessary and sufficient conditions for the memories and the origin to be stable, in terms of the relation between parameters of the transfer function g_σ.
-*Basins of attraction*: with the same radius for all memories, positive (L^2 norm).
-*Spurious memories*: they exist. If a spurious state vanishes at some point, then its basin of attraction has zero radius ($\| \ L^2 \ \|$) and it is a saddle point of the dynamics.

We also discussed the validity of these results when applied to the discrete models of AM [6],[7]. Such application looks more natural for the model with graded response [7], as in this case the concepts of euclidean distance and directional derivatives remain valid, while in the discrete case [6] only some general properties (concerning stability and convergence to attractors) are preserved, maintaining anyhow

[3]Two norms $\| \cdot \|$ and $\| \cdot \|'$ on a vector space V are said to be *equivalent* norms if there exist positive real numbers c and d such that $c \| x \| \leq \| x \|' \leq d \| x \| \ \forall x \in V$.

qualitative similarity with the infinite dimensional system.

This approach can be useful for modelling in biology and neurophysiology. It retains all the stylized facts that made attractive the Hopfield neural network and its modifications, yet giving the possibility of modelling the brain cortex as a continuous space. In other words, it integrates two levels of continuity:

-Continuous response units, which was already present in [7] and permits description of relevant neural activity by firing rates, rather than merely by the presence or the absence of an individual spike.

-Continuous topology of the neural system, obtaining a model of AM that reconciles biological evidence of a continuum of the neural tissue with descriptions provided by discrete models inspired in Ising systems.

In addition, the results proved here can be useful, with their limitations (Sect. 7), when performing the reverse track of what we have done, i.e. reconsidering the discrete case through the knowledge of what happens if the state space is continuous.

References

[1] Dreyfus, G.: Neural Networks. Methodology and Applications. Springer (2005)

[2] Fodor, J. A.: The Modularity of Mind. Cambridge, MIT Press (1983)

[3] Glauber R. J.: Time-dependent Statistics of the Ising Model. Journal of Math. Phys. **4** (1963) 294–307

[4] Hebb, D. O.: The Organization of Behavior: A Neuropsychological Theory. New York, Wiley (1949)

[5] Hertz, J., Krogh, A. and Palmer, R. G.: Introduction to the theory of neural computation. Addison-Wesley, Redwood City (1991)

[6] Hopfield, J. J.: Neural networks and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. **79** (1982) 2554–2558

[7] Hopfield, J. J.: Neurons with graded response have collective computational properties like those of two-state neurons. Proc. Natl. Acad. Sci. **81** (1984) 3088–3092

[8] Little, W. A.: The Existence of Persistent States in the Brain. Mathematical Biosciences **19** (1974) 101–120

[9] Little, W. A.: Analytic Study of the Memory Storage Capacity of a Neural Network. Mathematical Biosciences **39** (1978) 281–290

[10] MacLennan, B. J.: Field Computation in Motor Control. In Self-Organization, Computational Maps and Motor Control, ed. by Pietro G. Morasso and Vittorio Sanguineti, Elsevier-North Holland (1997)

[11] MacLennan, B. J.: Field Computation in Natural and Artificial Intelligence. Information Sciences **119** (1999) 73–89

[12] Segura, E. C. and Perazzo, R. P. J.: Associative memories in infinite dimensional spaces. Neural Processing Letters **12** (2) (2000) 129–144

[13] Segura, E. C. and Perazzo, R. P. J.: Biologically Plausible Associative Memory: Continuous Unit Response + Stochastic Dynamics. Neural Processing Letters **16** (3) (2002) 243–257

Genetically Modified GMDH Method with Cloning

Marcel Jiřina[1] and Marcel Jiřina, jr.[2]

[1] Institute of Computer Science, Pod vodarenskou vezi 2,
182 07 Prague 8 – Liben, Czech Republic
marcel@cs.cas.cz
http://www.cs.cas.cz/~marcel/
[2] Faculty of Biomedical Engineering, Czech Technical University in Prague,
Zikova 4, 166 36, Prague 6, Czech Republic
jirina@fbmi.cvut.cz
http://www.fbmi.cvut.cz/kontakty/jirina-marcel/

(Paper received on June 11, 2007, accepted on September 1, 2007)

Abstract. The GMDH MIA algorithm is modified by the use of the selection procedure from genetic algorithms and including cloning of the best neurons generated to get even lesser errors. The selection procedure finds parents for a new neuron among already existing neurons according to fitness and with some probability also from network inputs. The essence of cloning is a slight modification of parameters of copies of the best neuron, i.e. a neuron with the largest fitness. The genetically modified GMDH network with cloning (GMC GMDH) can outperform other powerful methods. It is demonstrated on some tasks from the Machine Learning Repository.

1 Introduction

In this paper we solve difficult classification tasks by the use of the GMDH MIA, i.e. group method data handling multilayer iterative adaptive method, modified by selection algorithm common in genetic algorithms and by cloning the best neurons generated up to given a moment of the learning process. Thus we combine genetic algorithm from which we use selection operation and immune networks from which we use a notion of cloning.

The basis of our method is the standard GMDH MIA method described in many papers since 1971 by [1], [4], [5], [6], [7], [10] and many others.

The basic approach of the GMDH is that each neuron (node or unit) in the network receives input from exactly two other neurons with the exception of neurons representing the input layer. The two inputs, x and y are then combined to produce a partial descriptor based on the simple quadratic transfer function (The output signal is z):

$$z = a_1 + a_2x + a_3y + a_4x^2 + a_5y^2 + a_6xy\,, \tag{1}$$

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 29-37

where coefficients $a, .. f$ are determined by linear regression and are unique for each transfer function, i.e. each neuron. The coefficients can be thought of as analogous to weights found in other types of neural networks.

The network of neurons with transfer functions (1) is constructed as one layer at a time. The first network layer consists of functions of each possible pair of n input variables (zero-th layer) resulting in $n(n\text{-}1)/2$ neurons. The second layer is created using inputs from the first layer and so on. Due to exponential growing of a number of neurons in a layer, after finishing the layer, the limited number of the best neurons is selected and the others removed from the network.

In [3], a selection procedure from genetic algorithms is used for finding two parents for a new neuron instead of the systematic procedure in the standard GMDH MIA algorithm with apparent layered structure. Our approach uses a selection procedure from genetic algorithms, but there is no explicit layered structure. The network grows during learning one neuron at a time. No neuron is deleted during the learning process and in the selection procedure its chance to become a parent for a new neuron is proportional to its fitness. If a new neuron appears to be the best, its clones are generated.

Clones are inexact copies of the parent neuron, which was found to be the best neuron up to now generated. Inexact copies follow from the fact that having exact copy has no sense in GMDH networks, so some mutation process must be applied to get clones a little bit different from the parent neuron.

It is shown here that a new algorithm, especially cloning, allows tuning the GMDH neural network more effectively than it is possible in genetically optimized GMDH networks.

2 Genetically modified GMDH

Here we describe approaches which result from our constructing of genetically modified GMDH network with cloning.

2.1 The learning set

We assume an n-dimensional real valued input and a one-dimensional real valued output. The learning set consists of n+1 dimensional vectors $(x_i, y_i) = (x_{1i}, x_{2i}, \ldots x_{ni}, y_i)$, $i = 1, 2, \ldots N$ where N is the number of learning samples or examples. The learning set can be written in the matrix form

$$[X, Y].$$

The matrix X has n columns and N rows; Y is a column vector of N elements. In the GMDH the learning set is usually broken to two disjoint subsets, the construction (training) set or setup set and the so-called validation set. In the learning process the former one is used for setting up parameters of neurons of the newly created layer, the

latter for evaluation of an error of newly created neurons. Thus $N = N_s + N_V$, where N_s is the number of rows used for setting up the parameters of neurons (the training set), and N_V is the number of rows used for error evaluation during learning (the validation set).

2.2 Selection process in GMDH MIA

Hiasaat and Mort have published a very interesting and principally simple approach in 2004 [8]. Their method does not remove any neuron during learning. Thus it allows unfit individuals from early layers to be incorporated at an advanced layer where they generate fitter solutions. Secondly, it also allows those unfit individuals to survive the selection process if their combinations with one or more of the other individuals produce new fit individuals, and thirdly, it allows more implicit non-linearity by allowing multi-layer variable interaction. The GMDH algorithm is constructed in exactly the same manner as the standard GMDH algorithm except for the selection process. In order to select the individuals that are allowed to pass to the next layer, all the outputs of the GMDH algorithm at the current layer are entered as inputs in the algorithm. It was shown in [8] that this approach can outperform the standard GMDH MIA when used in the prediction of two daily currency exchange rates.

2.3 New genetically modified GMDH network algorithm

The standard quadratic neuron is an individual of the genetically modified GMDH network. Its parents are two neurons (or possibly one or two network inputs) from which two input signals are taken. A selection of one neuron or input as one parent and another neuron or input as the other parent can be made by the use of different criteria. In genetic algorithms in selection step there is a common approach that the probability to be a parent is proportional to the value of the fitness function. Just this approach is used here. The fitness is simply reciprocal of the mean error on the validation set.

Note that in the standard GMDH MIA algorithm all possible pairs of neurons from the preceding layer (or inputs when the first layer is formed) are taken as pairs of parents. The selection consists of taking a limited number of the best descendants, "children", while the others are removed after they have arisen and were evaluated. In this way, all variants of the GMDH MIA are rather ineffective as there are a lot of neurons generated, evaluated and then simply removed with no other use.

An operation of a crossover in the genetically modified GMDH is, in fact, no crossover in the sense of combining two parts of parents' genomes. In our approach eq. (1) gives an exactly symmetrical procedure of mixing the parents' influence but not their features, parameters.

2.4 Selection procedure

The initial state form n inputs only, there are no neurons.

If there are k neurons already, the probability of a selection from inputs and from neurons is given by

$$p_i = n/(n + k) ,$$

$$p_n = k/(n + k)$$

for $n/(n + k) > p_0$, where p_0 is minimal probability that one of network inputs will be selected; we found $p_0 = 0.1$ as optimal. Otherwise

$$p_i = p_0,$$

$$p_n = (1 - p_0).$$

The fitness function is equal to the reciprocal error on the verification set. Let $\varepsilon(j)$ be the mean error of the j-th neuron on the validating set. The probability that neuron j will be selected is the following:

$$p_n(j) = (1 - p_n)\frac{1/\varepsilon(j)}{\sum_{s=1}^{N_T} 1/\varepsilon(s)}$$

Moreover, it must be assured that the same neuron or the same input is not selected as a second parent of the new neuron.

After the new neuron is formed and evaluated it can become immediately a parent for another neuron. Thus the network has no explicit layered structure. Each new neuron can be connected to any input or up to now existing neurons.

The computation of six parameters $a, .. f$, see (1), of the new neuron is the same as in the GMDH MIA algorithm.

2.5 Best neuron

A new neuron added need not be better than any other. Therefore, the index and error value of the best neuron is stored as long as a better neuron arises. Thus every time there is information about the best neuron, which can act as network's output. After learning, this output is used as a network output in the recall phase.

2.6 Pruning

After learning, the best neuron and all its ancestors have their roles in the network function. All others can be removed.

3 Cloning

There are various mechanisms or processes in the immune system, which are investigated in the development of artificial immune systems. A comprehensive summary can be found in [11].

Note first that authors dealing with artificial immune systems, e.g. [12] use rather different terminology than that used in neural networks community and genetic algorithms community. So some translation or mapping is needed. Here especially, antibody – neuron, affinity – fitness.

3.1 Cloning mechanisms

There are lots of ideas about approaches to cloning. From these ideas we use cloning in form close to the SIMPLE CLONALG algorithm [12] in this way:

```
BEGIN
      Given the Best GMDH Neuron with parents (i.e. input
      signals from) In1, In2 and with six parameters a, b,
      .. f.
      REPEAT
            Produce a clone of the Best Neuron. A clone
            has the same inputs In1 and In2 but mutated
            parameters a, .. f, i.e. parameters slightly
            changed (details see below).
            Evaluate fitness of this clone neuron.
            If this neuron happens to be better than the
            Best Neuron, break this clone generating cycle
            (and start this cloning algorithm from the
            beginning with new Best Neuron again).
      UNTIL a terminal criterion is satisfied or the
      maximum number of clones is reached.
END.
```

3.2 Mutation

It has no sense for the clones to be exact copies of the Best Neuron. Therefore, some mutation must be in effect. The clone to be a true clone must have the same parents. So, basic parameters – the two parents are not changed. A problem is how to change parameters $a, .. f$. These changes should be small enough to keep sufficient similarity of clone to original individual (the Best Neuron) and, at the same time, sufficiently large to reach necessary changes for searching data space in the neighborhood of parameters of the Best Neuron.

2.4 Selection procedure

The initial state form n inputs only, there are no neurons.

If there are k neurons already, the probability of a selection from inputs and from neurons is given by

$$p_i = n/(n + k) ,$$

$$p_n = k/(n + k)$$

for $n/(n + k) > p_0$, where p_0 is minimal probability that one of network inputs will be selected; we found $p_0 = 0.1$ as optimal. Otherwise

$$p_i = p_0,$$

$$p_n = (1 - p_0).$$

The fitness function is equal to the reciprocal error on the verification set. Let $\varepsilon(j)$ be the mean error of the j-th neuron on the validating set. The probability that neuron j will be selected is the following:

$$p_n(j) = (1 - p_n)\frac{1/\varepsilon(j)}{\sum_{s=1}^{N_T} 1/\varepsilon(s)} .$$

Moreover, it must be assured that the same neuron or the same input is not selected as a second parent of the new neuron.

After the new neuron is formed and evaluated it can become immediately a parent for another neuron. Thus the network has no explicit layered structure. Each new neuron can be connected to any input or up to now existing neurons.

The computation of six parameters $a, .. f$, see (1), of the new neuron is the same as in the GMDH MIA algorithm.

2.5 Best neuron

A new neuron added need not be better than any other. Therefore, the index and error value of the best neuron is stored as long as a better neuron arises. Thus every time there is information about the best neuron, which can act as network's output. After learning, this output is used as a network output in the recall phase.

2.6 Pruning

After learning, the best neuron and all its ancestors have their roles in the network function. All others can be removed.

neurons generated for stopping computation was 10000. Probability that a new neuron's input is one of input signals was 10 %, probability that a new neuron's input is one of already existing neurons was 90 %. The maximal number of clones generated from one parent neuron is limited to int(sqrt(No. of neurons generated up to now)). For all methods optimal threshold□ for a minimal error was used.

Classification errors for four different methods including the GMC GMDH are summarized in Table 1. In the second column the cross validation factor is given. The methods for comparison are 1-NN, the standard nearest neighbor method. Sqrt-NN, the k-NN method with k equal to the square root of the number of samples of the learning set. Bayes, the naïve Bayes method using ten bins histograms. LWM1, the learning weighted metrics method [14] modified with nonsmooth learning process.

Table 1. Classification errors for four methods on some data sets from UCI MLR.

	Algorithm				
Data set	1-NN	sqrt-NN	Bayes	LWM1	GMC GMDH
Shuttle-small	0.00259	0.00828	0.01294	0.00310	0.00259
Brest CW	0.04792	0.03255	0.05244	0.04538	0.04188
Vote	0.10526	0.06015	0.09774	0.07407	0.06667
Spam	0.09967	0.11270	0.14267	0.10735	0.10085
Adult	0.20826	0.21242	0.16370	0.17171	0.15923
Splice	0.40351	0.37206	0.28655	0.25874	0.13087
German	0.40767	0.20283	0.29768	0.72844	0.29465

In Table 1 and in Fig.1 it can be seen that the GMC GMDH method outperforms other methods in tasks Adult, Shuttle small, and Splice or nearly outperforms Brest CW, Spam, and Vote being the second best with very small difference with respect to the best method considered. It is the second best in task German.

5 Conclusions

The Genetically modified GMDH method is an elegant idea how to improve the efficiency of the popular GMDH MIA method. It is based on the usage of a selection principle of genetic algorithms instead of systematic assignment of all pairs formed by neurons of the last layer. Thus all neurons once generated remain at least potential parents for new neurons during the whole learning process. Also each input signal may be used with some probability as a parent signal for a new neuron. Thus the layered structure of the GMDH algorithm disappears because any new neuron can be connected to the output of any already existing neuron or even to any input of the network.

The target of this paper is to simplify the idea of a genetically modified GMDH neural network and to extend it by cloning. Clones are close but not identical copies of original individuals. An individual in our case is the best neuron. Intuition behind says

that even when the parameters of the best neuron were set up by linear regression, i.e. with a minimal mean squared error, due to nonlinearity of the problem as well as the GMDH network, the statistical normality assumptions are not met. Thus true minimum may lie somewhere in the neighborhood of parameters of the best neuron. Therefore the clones have mutated, i.e. slightly changed parameters of the parent individual, the best neuron. We found that the cloning with large parameters changes has small effect, but with small changes a new best neuron often arises.

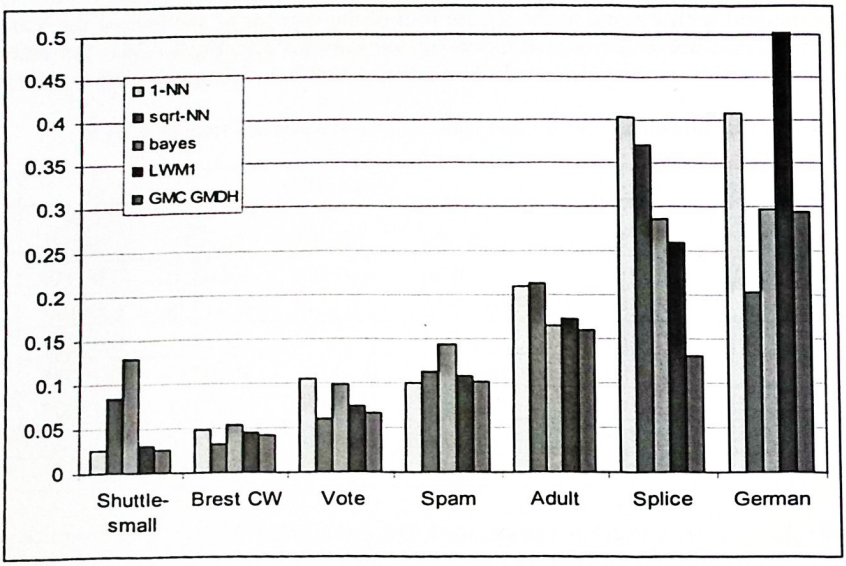


Fig. 2. Classification errors for four methods on some data sets from the UCI MLR. Note that for Shuttle small data the errors are ten times enlarged in this graph.

The genetically modified GMDH method with cloning (GMC GMDH) presented here appears to be a relatively simple and efficient method giving reliably good results better or comparable with the best results obtained by other methods. The new method behaves rather well and it has no critical parameters to be tuned. As there is no searching or sorting like in the nearest neighbor-based methods, the GMC GMDH is much faster than methods mentioned especially for large learning sets.

Essential advantage of the genetically modified GMDH with cloning is that one need not set up the number of the best neurons selected in the newly generated layer and thus indirectly control learning time and size of the network as in a standard GMDH MIA algorithm. Only the limitations of the genetically modified GMDH method with cloning are learning time or memory size.

Acknowledgement

This work was supported by the Ministry of Education of the Czech Republic under project Center of Applied Cybernetics No. 1M0567, and project No. MSM6840770012 Transdisciplinary Research in the Field of Biomedical Engineering II.

References

1. Ivakhnenko, A.G.: Polynomial Theory of Complex System. IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-1, No. 4, Oct. 1971, pp. 364-378.
2. Farlow, S.J.: Self-Organizing Methods in Modelling. GMDH Type Algorithms. Marcel Dekker, Inc., New York, 1984.
3. Tamura, H., Kondo, T.: Heuristics-free group method of data handling algorithm of generating optimal partial polynomials with application to air pollution prediction. Int. J.Systems Sci., 1980, vol. 11, No. 9, pp. 1095-1111.
4. Ivakhnenko, A.G., Műller, J.A.: Present State and New Problems of Further GMDH Development. SAMS, Vol. 20, 1995, pp. 3-16.
5. Ivakhnenko, A.G., Ivakhnenko, G.A., Műller, J.A.: Self-Organization of Neural Networks with Active Neurons. Pattern Recognition and Image Analysis, Vol. 4, No. 2, 1994, pp. 177-188.
6. Ivakhnenko, A.G., Wunsch, D., Ivakhnenko, G.A.: Inductive Sorting/out GMDH Algorithms with Polynomial Complexity for Active neurons of Neural network. IEEE 6/99, 1999, pp. 1169-1173.
7. Nariman-Zadeh, N. et al.: Modelling of Explosive Cutting process of Plates using GMDH-type neural network and Singular value Decomposition. Journ. of material processes technology Vol. 128, 2002, No. 1-3, pp. 80-87.
8. Hiassat.M., Mort.N.: An evolutionary method for term selection in the Group Method of Data Handling. Automatic Control & Systems Engineering, University of Sheffield, www.maths.leeds.ac.uk/statistics/ workshop/lasr2004/Proceedings/hiassat.pdf.
9. Oh, S.K., Pedrycz, W.: The Design of Self-organizing Polynomial Neural Networks. Information Sciences (Elsevier), Vol. 141, Apr. 2002, No. 3-4, pp 237-258.
10. F.Hakl, M.Jirina, E.Richter-Was: Hadronic tau's identification using artificial neural network. ATLAS Physics Communication, ATL-COM-PHYS-2005-044, last revision: 26 August 2005, http://documents.cern.ch/cgi-bin/setlink?base=atlnot&categ=Communication &id=com-phys-2005-044.
11. Negative Selection Algorithms: From the Thymus to V-Detector. Dissertation Presented for the Doctor of Philosophy Degree. The University of Memphis, August, 2006.
12. Guney K., Akdagli A., Babayigit B.: Shaped-beam pattern synthesis of linear antenna arrays with the use of a clonal selection algorithm. Neural Network World, Volume 16 (2006), pp. 489-501.
13. Merz,C.J., Murphy,P.M., Aha,D.W.: UCI Repository of Machine Learning Databases. Dept. of Information and Computer Science, Univ. of California, Irvine, http://www.ics.uci.edu/~mlearn/MLrepository.html, 1997.
14. R. Paredes, E. Vidal, Learning Weighted Metrics to Minimize Nearest-Neighbor Classification Error. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 7, July 2006, pp. 1100-1110.

Polynomial Cellular Neural Networks for Implementing Semitotalistic Cellular Automata

Giovanni Egidio Pazienza[1], Eduardo Gomez-Ramirez[2], and Xavier Vilasís-Cardona[3]

[1] GRSI, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Quatre Camins 2, 08022 Barcelona (Spain), email: gpazienza@salle.url.edu
[2] LIDETEA, Posgrado e Investigación, Universidad La Salle, Benjamín Franklin 47, Col. Condesa, 06140 Mexico City (Mexico), email: egr@ci.ulsa.mx
[3] LIFAELS, Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Quatre Camins 2, 08022 Barcelona (Spain), email: xvilasis@salle.url.edu

(Paper received on July 02, 2007, accepted on September 1, 2007)

Abstract. In this paper we study the relation between Polynomial Cellular Neural Networks (CNNs) and semitotalistic Cellular Automata (CA). First, we show that a Polynomial CNN, even in its simplest version, is capable of dealing with the Game of Life, a Cellular Automaton equivalent to a universal Turing machine; second, we prove that when Polynomial CNNs are employed to implement (semi)totalistic CA, the resultant network is always stable.

1 Introduction

Cellular Neural Networks (CNNs) [1] are a combination of Neural Networks and Cellular Automata, since neurons exhibit only local connections. Their importance resides in the topological simplicity which allows a direct VLSI implementation [2], unlike other neural networks models. Unfortunately, this benefit is balanced by their restricted computational power, since one-layer space-invariant CNN cannot solve linearly non-separable problems. In order to overcome this drawback, some modifications to the standard model have been proposed like space-variant CNNs [3], multilayer CNNs [4], trapezoidal activation functions [5], piecewise-linear discriminant functions [6], and the CNN - Universal Machine (CNN-UM), a supercomputer whose computation core is a CNN [7].
Our research is devoted to find the simplest way to extend the capabilities of one-layer space-invariant CNNs without losing the advantages offered by the VLSI implementation. In particular, we put forward a novel CNN model, first introduced in [8], in which a polynomial term is added to the state equation of the standard CNN. This polynomial model is able to deal with both elementary non-linearly separable tasks, like the XOR operation [9], and real-life problems, like the epilepsy seizures prediction [10], and it still has a VLSI realization [11].
Traditionally, the research in this field has been oriented to the applications, and only in the very last years a number of theoretically results have been found. For instance, in [12] some sufficient conditions for the stability of the network are given, whereas in [13] it is demonstrated that, at least in their discrete-time

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 39-48

version, polynomial CNNs are equivalent to a universal Turing machine.
In this paper we demonstrate two results about the continuous-time (CT) polynomial CNN: first, we show that the simplest type of CT polynomial CNN is capable of universal computation; then, we prove that a CT polynomial CNN implementing a particular class (semitotalistic) of Cellular Automata is completely stable. These aspects are particularly significant because they help to define a link between CNNs and CA, allowing to unify the theory of both structures.
The paper is structured as follows: first, we introduce the mathematical model for the standard and the polynomial CNN; then, we introduce a Cellular Automaton called Game of Life, explaining why it is so important and how it is possible to obtain the adequate network to perform it; third, we focus on some theoretical aspects of the stability of polynomial CNNs; finally, we draw conclusions.

2 Mathematical model

2.1 Generalities about the continuous-time CNN

A two-dimensional Cellular Neural Network is composed of a regular grid of dynamical artificial neurons (cells) with local connections only. Each CNN cell C_{ij} is coupled locally only to those neighbor cells which lie inside a prescribed *sphere of influence* $S_{ij}(r)$ *of radius* r, where

$$S_{ij}(r) = \{C_{kl} : max(|k-i|, |l-j|) \leq r, 1 \leq k \leq M, 1 \leq l \leq N\}.$$

For our purposes we employ a Cellular Neural Network with $r = 1$, then C_{ij} is coupled only to its eight nearest neighbor cells, and with space invariant weights. The cells of a CNN are dynamical systems, then they have an input, an output and a state. In particular, in a continuous-time Cellular Neural Network, the state of the cell in position (i,j) is

$$\dot{x_{ij}} = -x_{ij} + A * Y + B * U + z \tag{1}$$

where the symbol $*$ indicates the convolution operation that is defined, for the cell in position (i,j), as

$$A * Y = \sum_{|k|\leq 1, |l|\leq 1} a_{kl}\, y_{i+k,j+l}$$

Note that this way of defining the convolution is peculiar to CNNs, and it differs from the one usually employed in the image processing field. In the Eq. (1), the terms U and Y are the input and the output of the network, respectively; finally, A and B are 3-by-3 matrices, and they both (together with the bias z) determine the behaviour of the network. The output of the cell is computed as

$$y_{ij} = f(x_{ij}) = \frac{1}{2}(|x_{ij}+1| - |x_{ij}-1|) \tag{2}$$

It can be proved that, under certain conditions, the output converges always to $+1$ o -1. When CNNs are used to process images, the first case is equivalent to a black cell, and the second case to a white cell.

2.2 Polynomial CNN model

It is possible to prove that one-layer space-invariant CNNs cannot solve non-linearly separable problems. However, this drawback can be overcome by adding a polynomial term $g(\cdot)$ to the CNN model of Eq. (1), obtaining

$$\dot{x_{ij}} = -x_{ij} + A * Y + B * U + z + g(U, Y) \tag{3}$$

The polynomial CNN was first introduced in [8], and some sufficient conditions for the stability of the network are given in [12]. In the simplest case, the term $g(U, Y)$ is a second degree polynomial with the following form

$$\begin{aligned} g(U,Y) &= \sum_{k=0}^{2} (P_k * U^k \cdot Q_k * Y^{2-k}) \\ &= (P_0 * 1_{3\times3} \cdot Q_0 * Y^2) + (P_1 * U \cdot Q_1 * Y) + (P_2 * U^2 \cdot Q_2 * 1_{3\times3}) \end{aligned} \tag{4}$$

where $1_{3\times3}$ is a 3-by-3 matrix in which all the elements are 1's, and (P_0, P_1, P_2, Q_0, Q_1, Q_2) are 3-by-3 matrices. To sum up, the state equation for a cell of the continuous-time polynomial CNN can be written as

$$\begin{aligned} \dot{x_{ij}} = &-x_{ij} + A * Y + B * U + z \\ &+ (P_0 * 1_{3\times3} \cdot Q_0 * Y^2) + (P_1 * U \cdot Q_1 * Y) + (P_2 * U^2 \cdot Q_2 * 1_{3\times3}) \end{aligned} \tag{5}$$

3 The Game of Life

3.1 The rules

One of the best known Cellular Automaton (CA) is the 'Game of Life' (GoL), and its importance comes from the fact that the GoL has the same computational power as a universal Turing machine [14]. It is a no-player game played on an infinite two-dimensional grid of square cells, and its evolution is determined only by the initial pattern, which is defined by the programmer. Each cell interacts with its 8 neighbors, and at any fixed time it can be either black or white. In each time step, the next state of each cell is defined by the following rules

- *Birth*: a cell that is white at time t becomes black at time $t+1$ only if exactly 3 of its eight neighbors were black at time t;
- *Survival*: a cell that was black at time t will remain black at $t+1$ if and only if it had exactly 2 or 3 black neighbors at time t.

3.2 Representation of the rules of semitotalistic CA

In the GoL the next state of a cell depends only on its present state and on the sum of its eight nearest neighbors. This kind of CA are called *semitotalistic* and their rules can be conveniently represented in a Cartesian system, as depicted in Fig. 1. As usual for CNNs, a black pixel is +1 and a white pixel is -1. Then, the

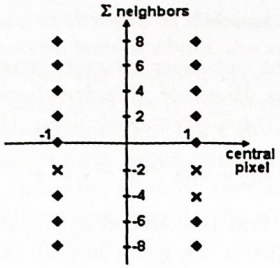


Fig. 1. The Game of Life: a red cross corresponds to a 1 (black) for the next state, and a blue diamond is a -1 (white).

x axis corresponds to the central pixel of the 3×3 Cellular Automaton, whereas the y axis is equal to the sum of the 8 outer neighbours. The rules of the CA are represented by associating to each point of the grid - corresponding to a pair (central pixel, sum of neighbours) - a symbol indicating the following state of the cell: a red cross is a 1 (black) for the next state, a blue diamond is a -1 (white). Thanks to this description, it is evident that the GoL is not a linearly separable task. These results can be summarized as in Table 1, showing that the 18 combinations are necessary and sufficient to describe the GoL.

4 A polynomial continuous-time CNN solving the GoL

4.1 Relation between polynomial CNNs and semitotalistic CA

Since in semitotalistic CA the next state of a cell depends only on its present state and on the sum of its eight nearest neighbors, the polynomial model of Eq. (5) can be simplified thanks to some considerations on the nature of the problem. First, all the matrices of the polynomial CNN model must have central symmetry: we indicate the central and the elements with the subindex c and p, respectively. Second, in general the output of a CA is a function of the input pattern only, thus all the matrices of Eq. (5) that are convoluted with Y and Y^2 - namely A, Q_0 and Q_1 - must have exclusively the central element. Finally, we focus on the Eq. (5): the terms $Q_2 * 1_{3\times3}$ and $P_0 * 1_{3\times3}$ are constant and they can be included into the matrices Q_0 and P_3, respectively; moreover, as the input of a CA is binary, the last term can be added to the bias because it is

$$Q_2 * U^2 = Q_2 * 1 = constant$$

In conclusion, a continuous-time polynomial CNN implementing a semitotalistic CA has the following form

$$\dot{x_{ij}} = -x_{ij} + q_{0c}\, y_{ij}^2 + (a_c + P_1 * U)\, y_{ij} + B * U + z \qquad (6)$$

Table 1. List of the 18 possible combinations for the GoL.

	Central pixel	# black neigh.	Output
1.	White	0	White
2.	White	1	White
3.	White	2	White
4.	White	3	Black
5.	White	4	White
6.	White	5	White
7.	White	6	White
8.	White	7	White
9.	White	8	White
10.	Black	0	White
11.	Black	1	White
12.	Black	2	Black
13.	Black	3	Black
14.	Black	4	White
15.	Black	5	White
16.	Black	6	White
17.	Black	7	White
18.	Black	8	White

where

$$B = \begin{pmatrix} b_p & b_p & b_p \\ b_p & b_c & b_p \\ b_p & b_p & b_p \end{pmatrix}, \; P1 = \begin{pmatrix} p_{1p} & p_{1p} & p_{1p} \\ p_{1p} & p_{1c} & p_{1p} \\ p_{1p} & p_{1p} & p_{1p} \end{pmatrix}.$$

This analysis allows to reduce the number of free parameters from 73 (9 parameters for each of the eight matrices appearing in the Eq. (5) plus the bias term) to only 7: $a_c, b_c, b_p, q_{0c}, p_{1c}, p_{1p}, z$.

4.2 CNN templates for the GoL

Finding the weights of a Cellular Neural Network that performs a given task is far from being trivial, and only partial solutions exist (e.g. [15, 16]). Nevertheless, a method for implementing the GoL on a discrete-time polynomial CNN is given in [13]. This technique, based on the solution of a system of equations representing the Game of Life, can be applied to the CNN model in Eq. (5) with only a few modifications [17]. The resulting parameters are

$$a_c = 40,\; b_c = 1.75,\; b_p = 1.75,\; i = 8,\; q_{0c} = -53,\; p_{1c} = 0,\; p_{1p} = -6$$

and the initial state of the network is $y_{ij}(0) = 0$. It is worth to mention that these value can be successfully found through a genetic approach too [18].
Now, it is necessary to check whether the proposed solution is stable according to the following definition, applicable to any autonomous dynamical system.

Definition 1 (System completely stable) *An autonomous dynamical system described by the state equation:*

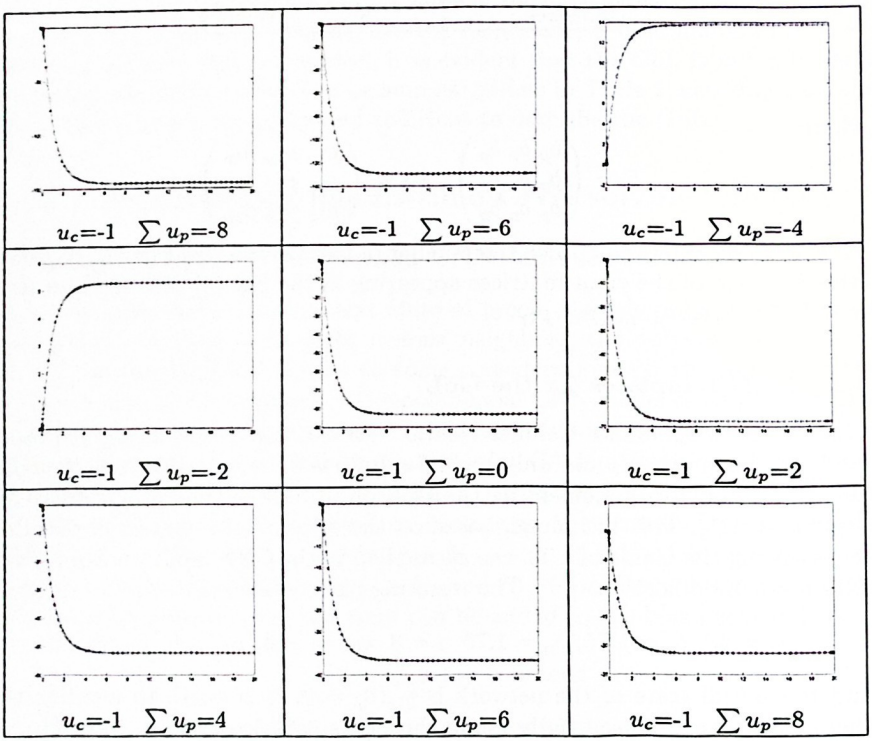
$$\dot{x} = F(x), \qquad x \in \Re^n, \qquad F : \Re^n \rightarrow \Re^n \tag{7}$$

is said to be completely stable if for each initial condition $x_0 \in \Re^n$

$$\lim_{t \rightarrow \infty} x(t, x_0) = const \tag{8}$$

The results of the simulations, summarized in Table 2 and 3 for all the 18 possible configurations of the input (see Table 1), indicate that the network is stable for the parameters and the initial state given, and that the output, calculated according to the Eq. (2) converges always to the desired values.

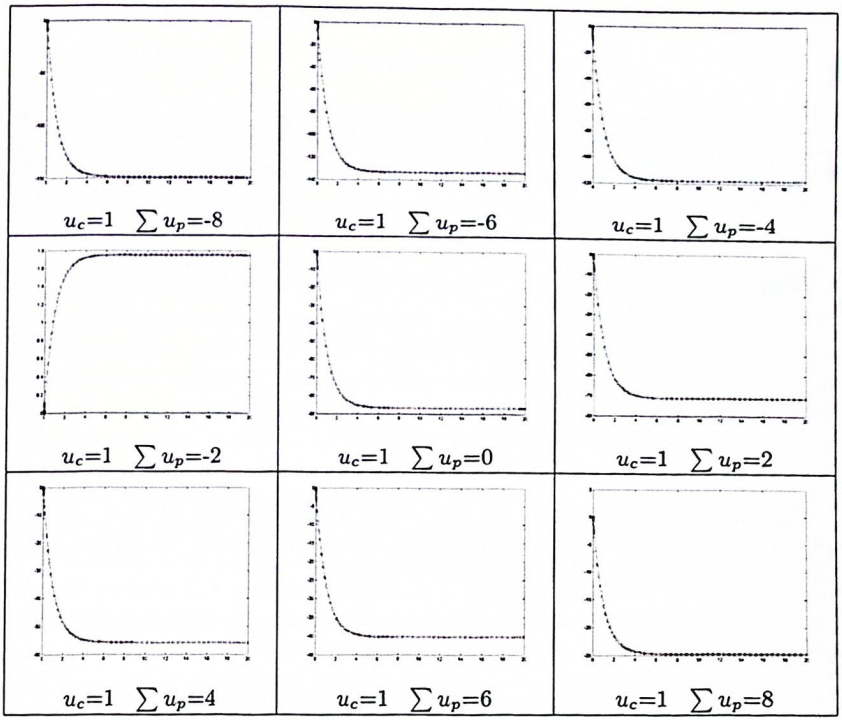
Table 2. Polynomial CNN described by the template of section 4.2. Time waveform for the state of a generic cell when u_c=-1.



5 About the stability of polynomial CNNs

In the previous section we saw that the network proposed to solve the GoL is stable, but this result is not immediately generalizable. Sufficient conditions for the stability of polynomial CNNs can be found in [12], but none of them is applicable to our network. Therefore, we need to provide a formal demonstration to show that the system described by the Eq. (6) is completely stable.

Table 3. Polynomial CNN described by the template of section 4.2. Time waveform for the state of a generic cell when u_c=1.



First of all, we mention a lemma that is useful in the prosecution of the work.

Lemma 1 (Gronwall's Lemma) *Let $u(t)$ a continuously differentiable function in $[0,T]$ such that*

$$\dot{u} \leq f(t)u(t) + g(t) \tag{9}$$

where $f(t)$ and $g(t)$ are integrable functions in $[0,T]$, then

$$u(t) \leq u(0)e^{\int_0^t f(\tau)d\tau} + \int_0^t g(\tau)e^{\int_\tau^t f(s)ds}d\tau \tag{10}$$

Proof. See [19]. □

The Gronwall's Lemma is necessary to proof the following theorem.

Theorem 1 (State-Boundedness Criterion) *If the function $f(\cdot)$ in the output equation (2) is continuous and bounded, then the state $x_{ij}(t)$ of each cell of a continuous-time polynomial CNN is bounded for all bounded threshold z and bounded inputs U.*

Proof. The proof can be derived from the one of a similar theorem for the Chua-Yang model presented in [20]. The Eq. (6) can be recast into the form

$$\dot{x}_{ij} = -x_{ij} + h(t) \tag{11}$$

where

$$h(t) \equiv q_{0c}\, f(x_{ij})^2 + (a_c + P_1 * U)\, f(x_{ij}) + B * U + z \tag{12}$$

Since both z and U are bounded by hypothesis, there exists finite constant K such that

$$\max_{0\leq t\leq\infty} |h(t)| \leq K \tag{13}$$

It follows from Eqs. 11 and 13 (via Gronwall's Lemma) that

$$\begin{aligned} |x_{ij}(t)| &\leq \left|x_{ij}(0)e^{-t}\right| + \left|\int_0^t e^{-(t-\tau)}h(\tau)d\tau\right| \\ &\leq |x_{ij}(0)|\, e^{-t} + \max_{0\leq t\leq\infty} |h(\tau)| \int_0^t e^{-(t-\tau)}d\tau \\ &< |x_{ij}(0)| + K, \text{for all t} > 0 \,\square \end{aligned}$$

Thanks to this theorem, we can assert that the state of each cell of the network is bounded, but the convergence to a certain value is still not assured. Now, we need to make use of another theorem.

Theorem 2 *Let $f : \Re \to \Re$ be a bounded continuously differentiable function, then every solution of $\dot{x} = f(x)$ is monotone.*

Proof. Suppose that $x(t)$ is a solution defined on an interval I; then, x is continuously differentiable on I. Suppose that $x(t)$ is not monotone too; then, there exist $t_1 < t_2 < t_3$ with $x(t_1) = x(t_3) \gtrless x(t_2)$. Without loss of generality, assume $x(t_2) > x(t_1)$. We may also assume $t_2 = \min\{t : t > t_1 \text{ and } x(t) = x(t_2)\}$, and $t_1 = \max\{t : t < t_2 \text{ and } x(t) = x(t_1)\}$; therefore, $x(t_1) < x(t) < x(t_2)$ for $t_1 < t < t_2$. By the Mean Value Theorem, there is t_4 with $t_1 < t_4 < t_2$ and $\dot{x}(t_4) > 0$. If $z = x(t_4)$, we have $x(t_1) < z < x(t_2)$, and $f(z) > 0$. Now, there must be t_5 with $t_2 > t_5 > t_3$ and $x(t_5) = z$, and we may take $t_5 = \min\{t : t > t_2$ and $x(t) <= z\}$. But since $x(t_5 - \delta) > x(t_5)$ for $\delta > 0$ sufficiently small, we must have $\dot{x}(t_5) \leq 0$, contradicting $\dot{x}(t_5) = f(x(t_5)) = f(z) > 0$. □

Note that the Theorem 2 can be applied to the Eq. (6), where

$$f(x_{ij}) = -x_{ij} + q_{0c}\, y_{ij}^2 + (a_c + P_1 * U)\, y_{ij} + K,$$

and K is a constant. Strictly speaking, this function is not continuously differentiable because of the form of the output function in the Eq. (2), but it can be approximated arbitrarily closely by a continuously differentiable function. To sum up, in our case for bounded threshold z and bounded input U, the state of each cell is bounded (Theorem 1) and monotone (Theorem 2). The convergence of the state for each cell of the network is assured by the existence of limits for monotone bounded functions (the well-known Monotone Convergence Theorem). Therefore, we can state that whatever the values for $a_c, b_c, b_p, q_{0c}, p_{1c}, p_{1p}, z$, the polynomial CNN representing a semitotalistic CA will be stable.

6 Conclusions

The fact that a continuous-time polynomial CNN can deal with the Game of Life allows us to state that it has, at least theoretically, the same computational power as a universal Turing machine. This is the first step towards the definition of a polynomial CNN universal machine capable of executing complex algorithms, similarly as the CNN-UM.

The second conclusion we can draw is that when polynomial CNNs are used to implement semitotalistic CA, the resultant network is always stable. This aspect is particularly important because it means that any possible choice of the CNN weights gives place to a valid semitotalistic Cellular Automaton. However, we cannot assert anything about the inverse implication, or rather, we have not proved yet that any semitotalistic CA can be implemented by using a polynomial CNN. This property is also useful when we determine the CNN weights necessary to perform a given task. Often such a problem, commonly called CNN learning, is solved by means of a genetic approach: in this case, the completely stability of the network assures that no unstable individual will be produced.

One of our long-term objectives is to discover an exact relation between the rules of a semitotalistic CA and the weights of a polynomial CNN. This would allow to transfer the knowledge about one structure to the other one, permitting to expand the field of application for both. A further aim is to test the polynomial model on complex algorithms, analysing whether it outperforms standard CNNs in terms of computational complexity.

Last but not least, we intend to implement soon the polynomial CNN model on a Field Programmable Gate Array (FPGA) in order to make possible a number of practical applications, especially in the image processing field.

References

1. L. Chua and L. Yang, "Cellular neural networks: theory," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1257–1272, Oct. 1988.

2. L. Yang, L. Chua, and K. Krieg, "VLSI implementation of cellular neural networks," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'90)*, vol. 3, May 1–3, 1990, pp. 2425–2427.
3. M. Balsi, "Generalized CNN: potentials of a CNN with non-uniform weights," in *Proc. second IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA'92)*, Munich, Germany, Oct. 14–16 1992, pp. 129–139.
4. Z. Yang, Y. Nishio, and A. Ushida, "Templates and algorithms for two-layer cellular neural networks neural networks," in *Proc. of IJCNN'02*, vol. 2, May 2002, pp. 1946–1951.
5. E. Bilgili, I. Goknar, and O. Ucan, "Cellular neural networks with trapezoidal activation function," *International journal of Circuit Theory and Applications*, vol. 33, pp. 393–417, 2005.
6. R. Dogaru and L. Chua, "Universal CNN cells," *International Journal of Bifurcation and Chaos*, vol. 9, no. 1, pp. 1–48, 1999.
7. T. Roska and L. O. Chua, "The CNN universal machine: an analogic array computer," *IEEE Trans. Circuits Syst. II*, vol. 40, pp. 163–173, Mar. 1993.
8. R. Schonmeyer, D. Feiden, and R. Tetzlaff, "Multi-template training for image processing with cellular neural networks," in *Proc. of 2002 7th IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA'02)*, Frankfurt, Germany, July 22–24, 2002.
9. E. Gómez-Ramírez, G. E. Pazienza, and X. Vilasís-Cardona, "Polynomial discrete time cellular neural networks to solve the XOR problem," in *Proc. 10th International Workshop on Cellular Neural Networks and their Applications (CNNA'06)*, Istanbul, Turkey, Aug. 28-30 2006.
10. C. Niederhofer and R. Tetzlaff, "Recent results on the prediction of EEG signals in epilepsy by discrete-time cellular neural networks DTCNN," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS'05)*, 2005, pp. 5218–5221.
11. M. Laiho, A. Paasio, A. Kananen, and K. A. I. Halonen, "A mixed-mode polynomial cellular array processor hardware realization," *IEEE Trans. Circuits Syst. I*, vol. 51, no. 2, pp. 286–297, Feb. 2004.
12. F. Corinto, "Cellular nonlinear networks: Analysis, design and applications," Ph.D. dissertation, Politecnico di Torino, Turin, 2005.
13. G. E. Pazienza, E. Gómez-Ramírez, , and X. Vilasís-Cardona, "Polynomial cellular neural networks for implementing the game of life," in *Proc. International Conference on Artificial Neural Networks (ICANN'07)*, Porto, Portugal, 2007.
14. P. Rendell. (2006) A Turing machine in Conway's game life. Available: www.cs.ualberta.ca/ ˜ bulitko/f02/papers/tmwords.pdf.
15. L. Chua and P. Thiran, "An analytic method for designing simple cellular neural networks," *IEEE Trans. Circuits Syst.*, vol. 38, no. 11, pp. 1332–1341, Nov. 1991.
16. J. Nossek, "Design and learning with cellular neural networks," in *Proc. third IEEE International Workshop on Cellular Neural Networks and their Applications (CNNA'94)*, Rome, Italy, Dec. 18–21 1994, pp. 137–146.
17. H. Harrer and J. Nossek, "Discrete-time cellular neural networks," *International Journal of Circuit Theory and Applications*, vol. 20, pp. 453–467, 1992.
18. E. Gomez-Ramirez and G. E. Pazienza, "The game of life using polynomial discrete time cellular neural networks," in *World Congress of the Fuzzy Systems Associations (IFSA'07)*, Cancun, Mexico, June 18–21, 2007.
19. T. H. Gronwall, "Note on the derivative with respect to a parameter of the solutions of a system of differential equations," *Annals of mathematics*, vol. 20, 1919.
20. L. O. Chua, *CNN: a paradigm for complexity.* Singapore: World Scientific, 1998.

Chaos in a Single Recurrent Artificial Neuron

L. Torres-Treviño

Corporación Mexicana de Investigación en Materiales
Oceanía 190. CP 25290,
Saltillo Coahuila, México
ltorres@comimsa.com.mx

(Paper received on June 22, 2007, accepted on September 1, 2007)

Abstract. A simple artificial neuron can represent a very complex behavior, inclusive chaotic one. Apparently, it is possible to control this behavior by means of the parameters that constitute this artificial neuron. This paper presents a single recurrent neuron which can generate different periodic pattern of activation. The results indicate a variety of the behavior since a constant behavior, a cyclic one and a chaotic behavior.

1 Introduction

The brain is a complex structure and a complex behavior can be generated from this structure including recognition, short temporal and long temporal memory, complex coordination and learning [1, 2, 3, 4, 5]. It is desirable to generate a similar structure to emulate this complex behavior and control them to generate entities with superior abilities. Nowadays it is very difficult to build entities which can emulate the complex activities of the brain; how-ever, recurrent artificial neural network is an approach to this objective. Usually there are several studies about the behavior of complex recurrent neural networks; however, our study is focused to handle single neurons. It has been proposed structures with few neurons, commonly no more than four neurons. It is applied a recurrence to generate different periodic patterns of activation, inclusive chaos [6, 7, 8, 9, 10, 11].

A single neuron can be defined as a single processing element which it emulate the behavior of a natural neuron in a simplified form. It is presented a single recurrent neuron where the activation function is a Gaussian function. The recurrence is applied when the output of this neuron is used as an input. This simple feedback makes a neuron to exhibit some times a set of well-defined periodic patters including chaos. The two parameters of the activation function change the periodic patters of activation. This paper shows the behavior of this neuron when there is a change of these two parameters.

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 49-56

2 Neuron with a Gaussian activation function

The architecture of a single neuron is shown in several books where neural networks are introduced. It is make an analogy between a natural neuron and an artificial neuron to get a simplistic model (figure 1). The artificial neuron is called a processing element because makes simple operations to integrate the input signals and it is generated an output using an activation function. Usually no linear activation functions are used, like sigmoid and hyperbolic tangential activation function. In this work a Gaussian activation function is used instead. The Gaussian activation function has a bell-like form. This function has two parameters, the width of the bell where it is called lambda and the center of mass called cm (equation1). Both parameters control the behavior of the activation function and consequently the behavior of the neuron is altered.

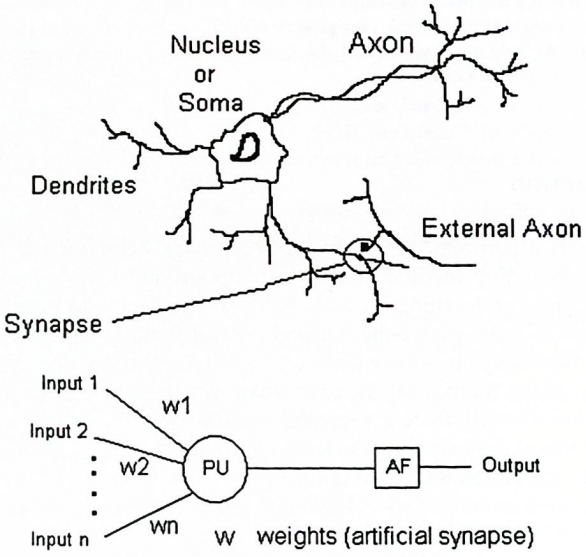


Fig. 1. There are some analogies between a natural neuron and an artificial neuron. Synapse is equivalent to a weight (W) and the activation output of the neuron through the Axon is determined by the activation function (AF).

$$Gauss(x.\lambda, cm) = e^{-(\frac{x-cm}{\lambda})^2} \tag{1}$$

The behavior of this single neuron can be modified including a feed-back of the output to the input. Figure 2 shows the recurrent neuron. The use of a Gaussian activation function is the difference with the artificial neuron shown in figure 1. The imple-

mentation of the model is shown where it is generated every pattern through a time unit t.

Step 1: Initialization: oa=0; epoch = 50;
Step 2: Define parameters Lambda and cm
Step 3: Calculate on = gauss (oa, lambda, cm);
Step 4: oa = on;
Step 5: if epoch <= 0 then end; else epoch <- epoch – 1; go to step (3)

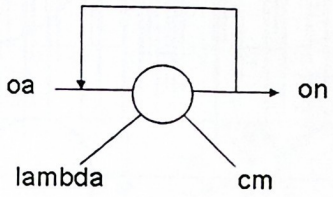


Fig. 2. Single recurrent neuron.

Specific parameters makes that the recurrent neuron exhibit a well defined periodic pattern (figure 3). Other parameters makes that the recurrent neuron exhibits a chaotic pattern (figure 4) and finally there are parameters that make a stable exhibition of the recurrent neuron (figure 5). Chaos is a complex behavior so it is necessary to elaborate a bifurcation diagram to show the possible fixed points of this recurrent neuron as a function of parameters of its activation function [12].

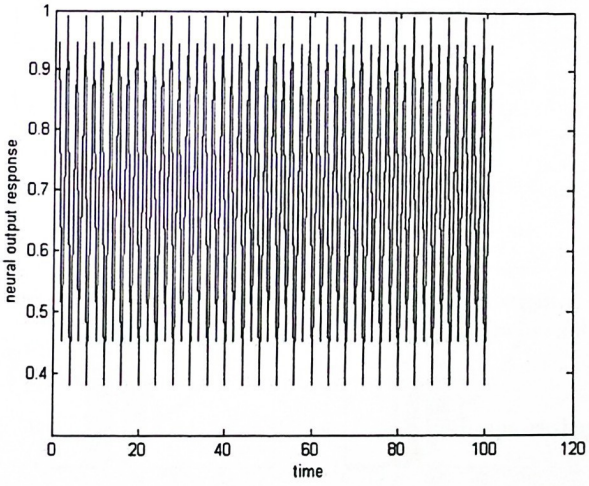


Fig. 3. Periodic pattern of a single neuron with Gaussian activation function (lambda = 0.5, cm =0.5).

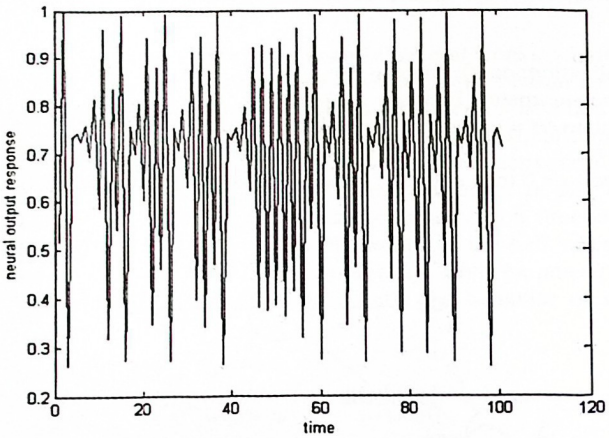


Fig. 4. A periodic pattern of a single neuron with Gaussian activation function (lambda = 0.43.

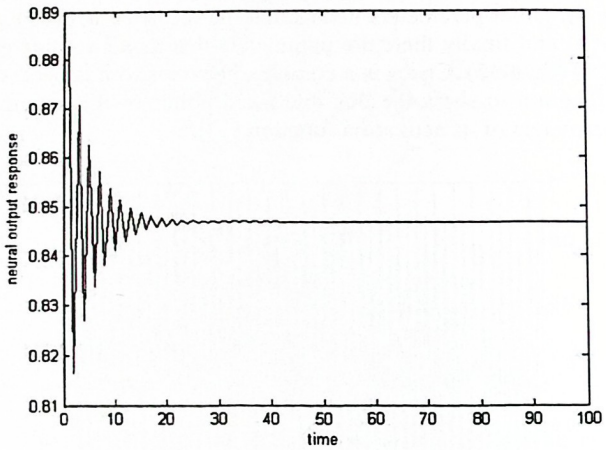


Fig. 5. One A periodic pattern of a single neuron with Gaussian activation function (lambda = 0.85, cm =0.5).

3 Description results

To illustrate a complete behavior of the neuron it is necessary to generate a bifurcation diagram common used when there is a chaotic behavior. In this diagram it is plotted a change of one parameter (i. e. lambda) and the other parameter is set fixed. All the

possible values of the neuron activation are plotter in y axis versus the changing parameter. It is used a normalized information, this means between 0 and 1.

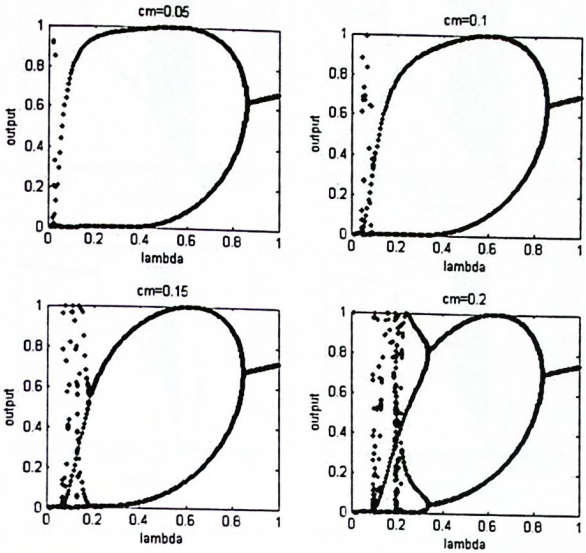


Fig. 6. One Bifurcation diagram of a single recurrent neuron using parameter cm from 0.05 to 0.2.

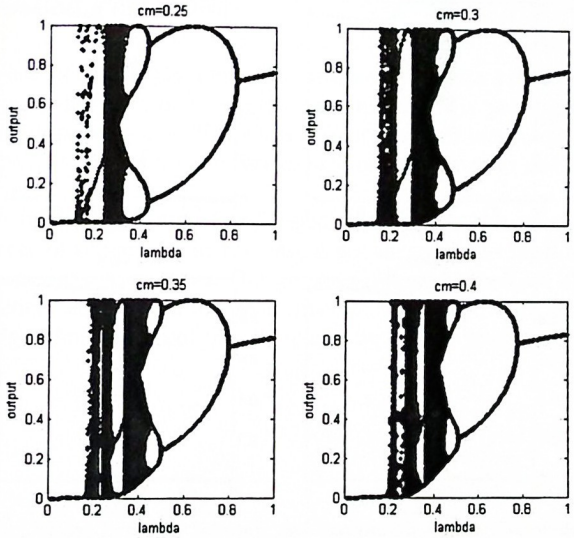


Fig. 7. Bifurcation diagram of a single recurrent neuron using parameter cm from 0.25 to 0.4.

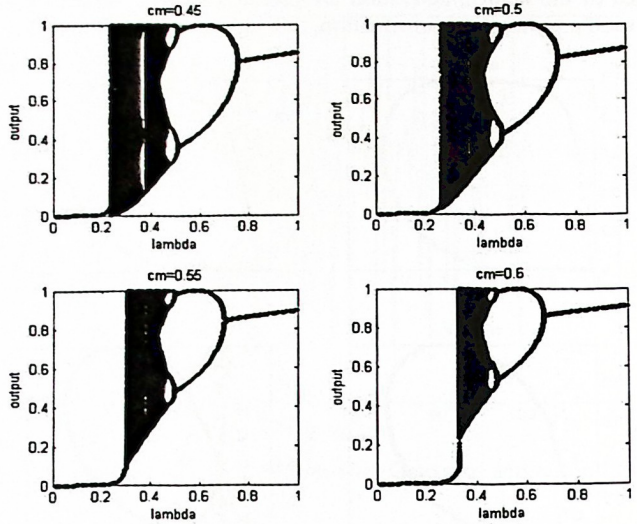


Fig. 8. Bifurcation diagram of a single recurrent neuron using parameter cm from 0.45 to 0.6.

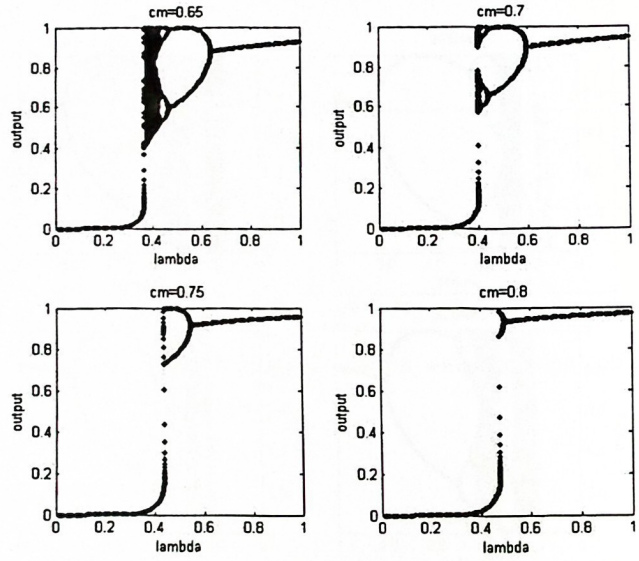


Fig. 9. Bifurcation diagram of a single recurrent neuron using parameter cm from 0.65 to 0.8

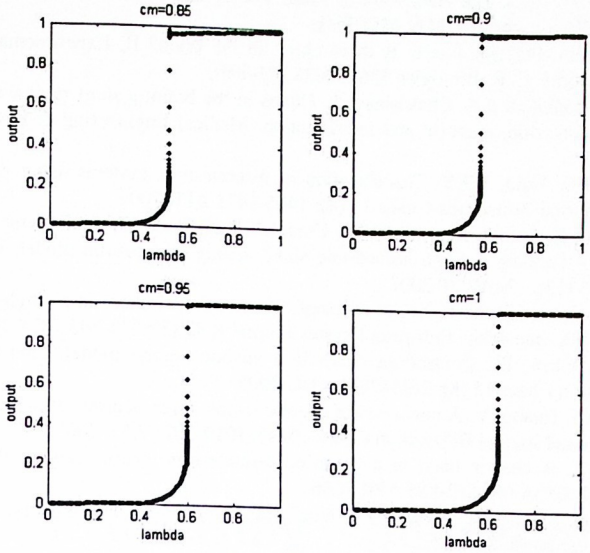


Fig. 10. Bifurcation diagram of a single recurrent neuron using parameter cm from 0.85 to 1.0.

4 Discussion and Conclusion

The interaction of several neurons of this kind could express a diversity of periodic patterns activities with a set of controllable parameters. It is possible to generate a variety of periodic patterns activities when it is used a rich interconnection of these neurons; however, it is necessary to realize new investigations to control all the parameters involved.

There is an implicit property of self organization where it is possible to store a complex sequence of actions and its response is not altered by the input sequence; that means, there is an attractor presence. This property is desirable because it is possible to store several action sequences in the same structure. The future work includes the use of a genetic algorithm to control this parameter an exhibit several desirable patters activities.

References

1. Erick R. Kandel and James H. Schwartz and Thomas M. Jessell. Essentials of neural science and behavior. Appleton & Lange. Simon & Shuster International Group. 1995.
2. Jean-Pierre Changeux, M. B. DeBevoise. The Physiology of Truth: Neuroscience and Human Knowledge (Mind/Brain/Behavior Initiative) Ed. Odile Jacob, 2002.

3. M.A. Arbib (Ed.), The Handbook of Brain Theory and Neural Networks, Second Edition (pp. 87-90). Cambridge, MA: MIT Press.
4. Henri Korn, Philippe Faure. Is there chaos in the brain? II. Experimental evidence and related model. C. R. Biologies 326 (2003) 787–840.
5. S.N. Sarbadhikari a,K. Chakrabarty b. Chaos in the brain: a short review alluding to epilepsy, depression, exercise and lateralization. Medical Engineering & Physics 23 (2001) 445–455.
6. Kwok, HS, Tang, WKS. Chaotification of discrete-time systems using neurons International Journal Bifurcation Chaos 14 (4): 1405-1411 APR 2004.
7. Channell, P, Author, Reprint Author Channell Paul Channell, Paul, Cymbalyuk, G, et al. Origin of bursting through homoclinic spike adding in a neuron model. Physics Review Letter 98 (13): - MAR 30 2007.
8. Li, CG, Chen, GR, Liao, XF, et al. Hopf bifurcation and chaos in a single inertial neuron model with time delay. European Physics Journal B 41 (3): 337-343 OCT 2004.
9. Lin, W, Chen, TP. Controlling chaos in a chaotic neuron model. International Journal Bifurcation Chaos 15 (8): 2611-2621 AUG 2005
10. Yang, XS, Huang, Y. A new class of chaotic simple three-neuron cellular neural networks International Journal Bifurcation Chaos 16 (4): 1019-1021 APR 2006.
11. Hiroto, T. A chaotic burst in a modified discrete-time neuron model. IEICE T FUND ELECTR E89A (4): 882-886 APR 2006.
12. Ingraham, R.L. (1992) A Survey of Nonlinear Dynamics: Chaos Theory. Cited 15 times. Singapore: World Scientific.

Image Processing Applications

Image Retrieval under Image Transformations and Occlusions

Roberto A. Vázquez and Humberto Sossa

Centro de Investigación en Computación – IPN
Av. Juan de Dios Batíz, esquina con Miguel Othón de Mendizábal
Ciudad de México, 07738, México.
Contact: ravem@ipn.mx, hsossa@cic.ipn.mx

(Paper received on July 25, 2007, accepted on September 1, 2007)

Abstract. An important problem in image management is image retrieval. When images in a database are not well organized, their efficient retrieval is a problem. In this paper we describe an image retrieval system able to recover a set of images that satisfy some searching criteria. The proposal works well even if the objects appearing in the images are occluded or suffer affine transformations as rotations and translations. The proposed system is composed of three modules. The first module performs object training and object recognition using associative memories as classification tool. The second module performs image analysis and image organization in a database. The third module allows for image retrieval. Through several experiments we show the efficiency of the system.

1 Introduction

Image retrieval is an important problem nowadays. In the WWW more that 73% of the information is images [8]. Images in this space are, in general, not well or not organized at all. The explosion of image databases and the inefficiency of text-based image retrieval have created an urgent need for effective approaches in image database retrieval. Their search is normally high time-consuming. Many techniques and commercial systems have been proposed in the literature to succeed this task. For several examples, refer to [1-5, 8-11]. Most of these systems use combinations of image-features such as colour or texture to organize the images as a database of images and then to recover them from it. To the final user of the system, normally these image features do not have any meaning or are difficult to interpret and use. To avoid this problem, some systems use images as examples to recover other similar images. For a system to be useful, it is necessary to be designed in accordance to the user's intuition. A person normally knows nothing about describing features, image processing or image analysis. A user of and image retrieval system would like to input directly to the system queries such as:

1. "System, display the set of images with rivers and threes", or
2. "System, show me the images with lions and zebras".

The key problem to most computer vision applications comprising image organization and retrieval is object recognition. Object recognition, in the general case, is still

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 59-68

an open problem and it strongly determines the functionality of many systems such as image retrieval systems. In [14] was described an image retrieval system called (SISREC). Given a set of images the system automatically organizes a database. This system was able to recover images from a database in terms of their objects. One of the main restrictions of SISREC was that the objects in the image were not in contact with any other object.

In this paper we describe an improved version of SISREC that is able to recover images now in the presence of object occlusions. The system is composed of three modules that are next described. Is important to mention that modules about to object recognition and image organization are the main contributions of this paper.

2 Description of the system

The system here proposed is composed of three main modules:

1. One of Object Recognition Module (ORM).
2. One of Image Analysis and Structuring Module (IASM), and
3. One of Image Retrieval Module (IRM).

2.1 Object recognition module (ORM)

This module is the most important of the three. Its performance determines the complete efficiency of the other two, and as a result that of the system. This demands using powerful object recognition techniques to get good results. Because of objects can appear occluded we decided to adopt some ideas described in [16] for object recognition under occlusions.

2.1.1. Describing essential parts of an object. For an object to be recognized in an image in the presence of occlusions we first detect its so-called essential parts (EP). An EP part is a part that allows finding out the presence of an object in an image. To detect an EP of an object, we get an image of the object with a background as homogeneous as possible. For an example refer to Figure 1. Then we continue as follows:

1. Manually, with a circular window we first select a region of the image enclosing a candidate EP the object.
2. Inside this window, we apply a standard threshold [12] to get a binary sub-image.
3. We apply to this binary circular image a connected component-labeling algorithm [6] to get all possible connected binary components.
4. We remove small spurious regions with a standard size filter [7].
5. We then calculate well-known first four Hu descriptors, invariant to translations and rotations, to describe locally the selected part. For the details refer to [13].
6. We apply steps 1 to 5 to other parts of the object.
7. We repeat this procedure (steps 1 to 6) to remaining objects.

Special attention has to be put in selecting the set of describing parts. For example, from Figure 1, we might think the head of the bolt and the hole of the washer could be

two EPs that will allow differentiation between these two objects. However by comparing the describing features from both parts we have noticed that they could be similar. To select EP or EPs by which we are going to perform object detection under occlusions, we have just taken from the list of pre-selected parts as that EP that allows more discrimination with other objects. For this we have used their corresponding describing features.

2.1.2. The classification tool. An associative memory is a mathematical device specially designed to recall complete patterns from inputs patterns that might be altered with noise. An associative memory M can be viewed as an output-input system as follows: $\mathbf{x} \rightarrow \mathbf{M} \rightarrow \mathbf{y}$, with $\mathbf{x}$ and $\mathbf{y}$, respectively the input and output patterns vectors. The structure of an associative memory for pattern classification is simply another way to see a neural network. In this work we use an extended associative memory useful to classify real-valued patterns, by assigning them to the class by their index.

(a) (b)

Fig. 1. (a) Head of bolt selected as its essential part. (b) Hole of washer selected as it essential part.

Let $\left(x^{\xi}, i\right)_{\xi=1}^{p}, x^{\xi} \in \Re^{n}, i = 1,\ldots,m$ a set of p fundamental couples (SFC), composed by a pattern and its corresponding class-index. The problem is to build an operator M, using these SFC, that allows classifying the patterns into their classes. This mean, $\mathrm{M} \otimes x^{\xi} = i$ for ξ=1,..., p and that even in the presence of distortions it classifies them adequately, that is $\mathrm{M} \otimes \tilde{x}^{\xi} = i$, where $\tilde{x}^{\xi}$ is an altered version of x^{ξ}. A first approach in this direction was presented in [15]. Operator $\otimes$ is chosen such that when operating vector x^{ξ} with matrix $\mathbf{M}$, produces as result the corresponding index class of pattern x^{ξ}.

Matrix $\mathbf{M}$ is build in terms of a function ϕ as follows:

$$\mathbf{M} = \begin{bmatrix} \phi_1 \\ \vdots \\ \phi_m \end{bmatrix} \tag{1}$$

Function ϕ is computing using so-called "sep" operator, which allows us transform a set of relatively close ϕ_i's, into another set of more separated ϕ_i''s. This transformation, as we will next see, allows improving the associative memory's performance. Function ϕ is defined as follows:

$$\phi_{ij} = \gamma_{ij} + \lambda_{ij} \tag{2}$$

where $\gamma_{ij} = \bigvee_{\xi=1}^{p} \left(x_j^{i\xi} \right)$ and $\lambda_{ij} = \bigwedge_{\xi=1}^{p} \left(x_j^{i\xi} \right)$.

Pattern classification is performed as follows. Given a pattern $x^{\xi} \in \Re^n$, not necessarily one of the already used to build matrix $\mathbf{M}_{sep}$, class to which x is assigned is given by:

$$i = \mathbf{M} \otimes x^{\xi} = \arg \underset{l}{\left[\bigwedge_{l=1}^{m} \bigvee_{j=1}^{n} \left| m_{lj} - r_{lj} \right| \right]} \tag{3}$$

where $r_{lj} = x_j + \gamma_{lj}$.

2.1.3. Building the associative memory for object recognition under occlusions. Once selected the essential feature of each object we would like to recognize, associative memory $\mathbf{M}$ is built as follows:

1. Obtain 20 images of each object in different positions and rotations
2. To each sub-image containing the selected EP, calculate the corresponding Hu's invariants as explained in section 2.1.1.
3. With these values, build corresponding associative memory as explained is Section 2.1.2.

Once trained the classification tool, the output of the ORM is the identity of a given object (from image-features to objects).

2.2 Image analysis and structuring model (IASM)

This module receives as input a set of *n* images containing one or more instances of the objects already learned by the system even under occlusions. To determine if an instance of an object is in an image we use so-called blocking swapping algorithm (BSA). This algorithm allows extracting information from an image to be used to operate the associative memory already trained. Information is extracted by means of a mask MA of size equal to the original image. This mask is filled with circular windows of ratio of 15 pixels, see Figure 2 (a). To avoid analysing several times the same region of the image, the BSA uses a blocking table. Blocking table is used to decide if a window is used or no, for an example refer to Figure 2 (b). Blocking table allows blocking those regions in the image that have been selected as regions containing the distinctive part of an object. Detection of the essential part of an object is performed in six steps as follows:

For each of the *n* images to be structured:

1. Clear blocking table.

2. Refresh mask **MA**.
3. Apply logical **and** operation between mask **MA** and image to be analysed.
4. Compute first four Hu invariants to each region enclosed by circular window in the new image.
5. Apply associative memory to each obtained describing vector and if vector corresponds to essential part of one of the objects we are looking for, then a vote is given for this object and block associated region in blocking table.
6. If whole image has been analysed then finish stop algorithm, else go back to step 2.

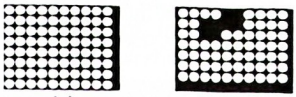


Fig. 2. (a) Mask used to subtract form image the information to train associative memory. (b) Mask with blocked regions.

The output of this module is a list of pointers from each object to the images that contain this object, organizing automatically the set of images in a table or in a database. In table form each locality of this table contains the evidence that a given object $O_i, i = 1, \cdots, q$ is contained in a given image $I_j, j = 1, \cdots, n$, see figure 3(a).

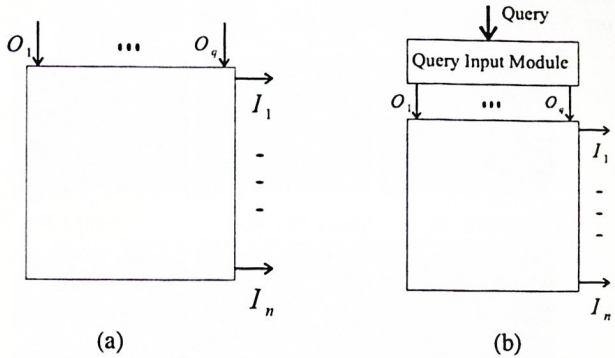


Fig. 3. (a)The output of the IASM Module is a table of pointers from objects to images. (b) The IRM module receives queries as inputs and outputs list of images satisfying these queries.

2.3 Image retrieval module

The input to this module is a query in terms of the type of objects an image may contain, see Figure 3(b). Its output is a list of the images satisfying the input query. Examples of queries this module accepts follow:

1. A=1 & C=1, means the system will display all images (if any) containing instances of object A and object C.

2. B=0 & D=1& E=0, means the system will display all images (if any) not containing instances of object B and E, and containing instances of object.

3 Experimental results

Performance of the system was tested with a set of five real objects: a bolt, a washer, an eyebolt, a hook and a dovetail. Images of them are shown in Figure 5. Twenty different images of each of these five objects were used to train the associative memory.

(a) (b) (c) (d) (e)

Fig. 4. The five objects used in the experiments. (a) A bolt. (b) A washer (c) An eyebolt (d) A hook, and (e) A dovetail.

One hundred images containing one or more instances of the objects shown in Figure 4 were used to train the classifiers and to organize these images in terms of theirs objects. Figure 5 shows nine of these images.

Image No. 21 Image No. 40 Image No. 56

Image No. 71 Image No. 75 Image No. 90

Image No. 95 Image No. 98 Image No. 100

Fig. 5. Nine of the images structured by the system.

With 5 objects and the operators "AND" and "OR", and the restriction that an object is asked to appear or not asked to appear in and image, the number of possible queries that can be input to the system is 1,652: $(A=1 \,\&\, B=1)$ is one combination; $(B=1 \,\&\, C=0 \mid E=1)$ is another combination.

Once this set of images were organized automatically for the system we test the accuracy of the system to recover images in terms of their objects using only 800 hundred queries (almost the 50% of whole possible queries).

The overall percentage of the system is of 83.0%. This percentage of performance was obtained as follows: Given a query, nt images are output by the system. In only nc of images the desired objects appear and in $nt - nc$ at least one object does not appear (does not satisfy the query). The partial performance percentage for this query is obtained as: nc / nt. By computing this partial percentage for each query and by summing-up the total of partial percentages and by dividing this total by the number of queries we get the total average performance percentage.

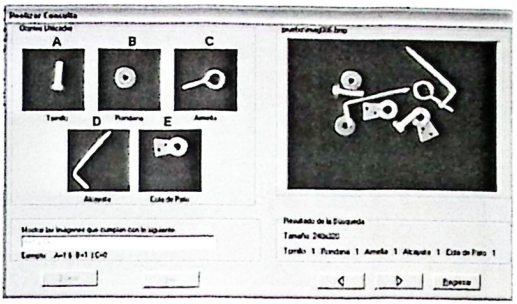


Fig. 6. Image 86 recovered by the system when query A=1 and C=1 is introduced to the system. 43 images were recovered by the system.

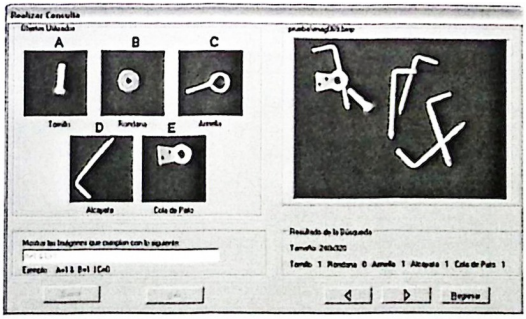


Fig. 7. With the set of 43 image recovered when query A=1 and C=1 is fed to the system, image 69 was recovered. Note there is no an instance of object C, however instances of object A do appear.

Figures 6 and 8 show two of output results. In the first case a set of 43 images were recovered by the system. As you can appreciate in Figure 6, image 86 is output by the system and satisfies the search criterion; only 23 % of recover images do not satisfied the search criterion, one example is given in Figure 7.

In the second case a set of 24 images were recovered by the system. As you can appreciate in Figure 8, image 96 is output by the system and satisfies the search criterion; only 12 % of recover images do not satisfied the search criterion, see for example Figure 9.

The worst results were obtained when the query involves object D. Next we will explain why the low accuracy is obtained when a query involves object D. The total performance of the system depends strongly on the functioning of the ORM. If the system is able to recognize accurately instances of the objects in an image, it can thus be inserted in the right position in the database. Otherwise an erroneous pointer to it will be created producing, of course, at the moment a query is input to the system an erroneous result. The ORM was tested isolated apart from the image retrieval system using the same set of images. The obtained classification results were acceptable. Table 1 summarizes these results.

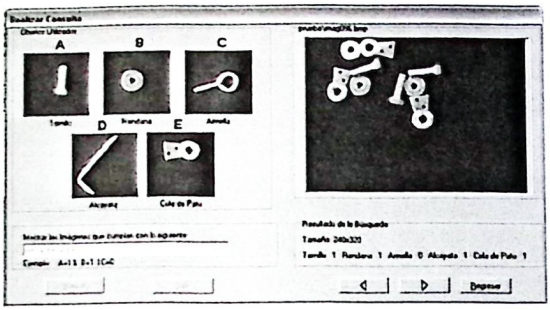


Fig. 8. Image 97 is recovered by the system when query A=1 and B=1 and E=1 is introduced to the system. 24 images were recovered by the system.

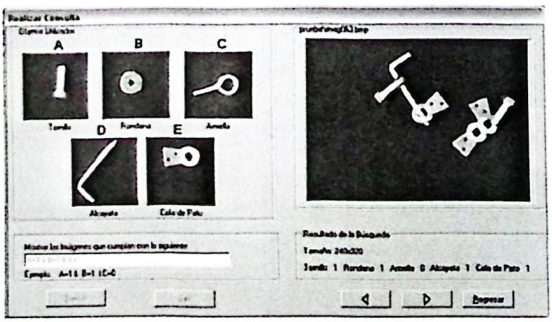


Fig. 9. With the set of 24 image recovered when query A=1 and B=1 and C=1 is injected to the system, image 63 was recover. Note there is not an instance of object B, however instances of object A and E exist.

As you can appreciate in Table 1, the worst classification result was obtained with the hook (Object D). This is because of the essential part selected for the object. This bad classification causes that retrieval system performs a low accuracy when the query involves object D.

4 Conclusion and directions for further research

In this paper we have described a system that is able to first organize a set of input images, and second to allow for recover examples of them given an image query. Three modules integrate the system. The first module performs object training and object recognition. The second module performs image analysis and image organization. The third one allows for image retrieval.

Table 1. Classification results obtained with the object recognition module.

Object	% of classification
Bolt	93%
Washer	92%
Eyebolt	80%
Hook	51%
Dovetail	96%
% total of Classification	**83%**

Through some experiments we have shown the efficiency of the system. The overall performance of the system strongly depends on the correct functioning of the object recognition module. For the set of objects used and the restrictions imposed the system provides good results.

One important feature is that it is close to the end-user that usually ignores everything about image processing and image analysis. It is a first step to design a more general system.

A system with these limitations could be used, for example, to recover photographs of objects circulating through a conveyor belt.

Nowadays, we are working on: 1) a method to obtain automatically the EPS for each object, 2) how to recognize object also in the presence of scale changes, and 3) how to solve the most difficult problem of image retrieval under clutter images. At the end of our research, we hope to count with a system able to structure a set of images (landscapes, portraits, and so on), allowing to recovering them in terms of their objects and their relations.

Acknowledgments. This work was economically supported by SIP-IPN under grant 20071438 and CONACYT under grant 46805.

References

1. A. Del Bimbo, and P. Pal (1997). Visual Image Retrieval by Elastic Matching of User Searches, IEEE Trans on Pattern Analysis and Machine Intelligence, 19(2):121-132.
2. A. Del Bimbo (1999). Visual Information Retrieval, Morgan Kaufmann Publishers, San Francisco, CA.
3. R. Egas et al. (1999). Adapting k-d Trees to Visual Retrieval, Proc. Visual 99, LNCS 1614: 533-540.
4. M. Flickner et al. (1995). Query by image and video content: The QBIC system, IEEE Computer, 28(9):23-32.
5. T. Gevers, and A. Smeulders (1997). PicToSeek: A Content-Based Image Search System for the World Wide Web, Proc. Visual 97, Chicago, pp. 93-100.
6. R. C. Gonzalez and R. E. Woods (2002). Digital Image processing, Second edition, Prentice Hall, Inc 2002.
7. R. Jain et al. (1995). Machine Vision, McGraw-Hill.
8. M. S. Lew (2000). Next-Generation Web Searches for Visual Content, IEEE Computer, 33(11):46-52.
9. M. S. Lew (2001). Principles of Visual Information Retrieval, Advances in Pattern Recognition Series, Springer-Verlag London Limited.
10. J. R. Smith and S. F. Chang (1997). Visually searching the web for content, IEEE Multimedia, 4(3):12-20.
11. L. Taycher, M. Cascia and S. Sclaroff (1997). Image Digestion and Relevance Feedback in the ImageRover WWW Search Engine, Proc. Visual 97, Chicago, pp. 850-91.
12. F. Jiulun and X. Winxin (1997). Minimum error thresholding: A note, Pattern Recognition Letters, 18(8):705-709.
13. M. K. Hu (1962). Visual pattern recognition by moment invariants, IRE Transactions on Information Theory, 8:179-187.
14. H. Sossa et al. (2003). SISREC: A System for Image Retrieval, In proceedings of Forth Mexican International Conference on Computer Sciences (ENC 2003), IEEE Computer Society, 69-73.
15. H. Sossa, R. Barrón, R. A. Vázquez (2004). Real-Valued Pattern Classification based on Extended Associative Memory, In proceedings of Fifth Mexican Conference on Computer Science (ENC2004), IEEE Computer Society 213-219.
16. R. A. Vázquez (2005). Invariant descriptions and associative processing applied to object recognition under occlusions, Master Thesis (In Spanish), CIC-IPN.

Cursive Character Recognition by combining Describing Features, Slalom Method and Daubechies Wavelet

L. K. Toscano, J. H. Sossa, R. Barron, G. Sanchez

Centro de Investigación en Computación – IPN
Av. Juan de Dios Batiz, esquina con Miguel Othón de Mendizábal
Ciudad de México, 07738, México.
Contact: likatome@hotmail.com and hsossa@cic.ipn.mx

(Paper received on July 10, 2007, accepted on September 1, 2007)

Abstract. In many applications, characters are written directly by a person. In these cases, the main problem is the wide variability of these characters. In a strict way we should consider them as random signals. To handle this variability many Artificial Intelligence based tools have been proposed, among them neural networks, support vector machines. Following a different strategy, in this paper, Splines are used to adjust to a finite set of samples obtaining as a result a representative pattern of the trace as an optimal set of the nodes of the Spline. We then use wavelets to decompose the initial samples of the trace into its components: approximation and detail. In this paper we consider only the approximation part due to for recognition purposes it is relevant. As we will see an enhancement is obtained by normalizing the optimal nodes. In the experiment section we show the entire enhancement obtained.

1 Introduction

The handwriting character movement is caused by an act of intentional muscular force and joints elasticity. In order to recognize handwritten characters, it is necessary to extract the features related to this movement. Conventional manuscript character recognition is based on the feature extraction from the character shape under analysis. These features can be the lines inclination, the relative position of each line, the length of the different parts of the line, and so on [1]. For example this approach can be used to recognize efficiently non-cursive characters. However, for cursive characters, this approach is not well suited.

One reason why we humans are able to read and understand cursive characters (very aerodynamic or deformed) is because somehow we have the ability to mentally trace several times the letter in the order it was written. When a person writes a character, generally realizes 4 steps, which are:

(a) In the mind, the person imagines the character symbol that he wants to write,
(b) His brain transmits the movement order to his muscle and joints,
(c) He realizes a series of movements according to character writing order,
(d) The image character is made in consequence of three steps (a) to (c).

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 69-80

The character generation process is made from the step (a) to (d), while the recognition process can be performed in inverse order, its means from step (d) to (a). However, the inverse process for character recognition is quite difficult.

As it is known, character recognition can be done off-line and on-line [1]. On-line character recognition requires the physical presence of the person that writes the character. Features used in this case are the pen pressure, the trace speed, the trace directions sequence and others [1]. On-line character recognition demands the realization of steps (c) to (a), while off-line character recognition is equivalent to the inverse realization of all steps from (d) to (a). Thus off-line character recognition is a part of the complete inverse on-line manuscript character recognition process. In the past time several manuscript character on-line recognition systems have been proposed. For example, in [2] the authors used a neural network to recognize cursive isolated characters. In [3] a study where the Laplace transform and a second order lineal model that takes the writing velocity as a variable control was used to synthesize the inverse process.

In [4], the authors combined well-known HHMs and dynamic programming for the cursive characters recognition. Character segmentation and its recognition are performed by this combination, 91% efficiency recognition for the English characters was obtain.

In this paper, the main proposal is focused on the inverse writing realization steps from (c) to (a). Steps (c) and (b) are based on the approximation given by a Spline function which it is possible to obtain the movement order, required to perform the character trace using a digitizing tablet. The process from step (b) to (a) is carried out likening the generated models for each character; finally a tree layer neural network is used to train the feature vectors from the optimal knots.

2 Proposed System

The proposed system consists in a feature extraction and recognition stages respectively. In the feature extraction stage, the optimal knots sequence for each character are obtained, which are significant points of the handwritten character. Then using these points, the handwritten character shape can be reconstructed [4]. In this stage, a natural Spline function (SLALOM method) and Steepest Descent method are applied repeatedly until with 20 knots be sufficient to reconstruct the character with minimum error. These 20 optimal knots are used as a feature vector in the recognition stage, using a three-layer Backpropagation neural network. Figure 1 shows the proposed system structure.

2.1 Data acquisition

Data from each character is obtained by means of a digitizing tablet. For writing an ergonomic pencil Intous 2 of Wacom was used. With this pencil people was asked to write the characters on the tablet. This allows knowing the order of articulation of each character. From the tablet we can get an image of the written character and the data as

how the character was written. Figure 2 shows, for example, the data acquired for character "h" and its signal in the x and y axes.

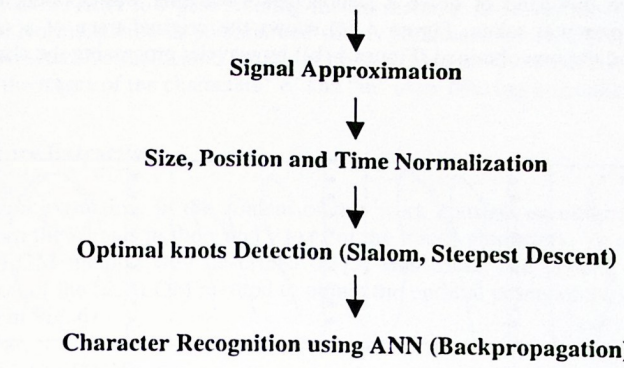


Fig. 1. Proposed system structure

2.2 Database Construction

To build the database of characters to be recognized, the 26 letters of the English alphabet were used. For each character 50 samples were obtained. In the case of this research, all samples were obtained from one writer. The database contains thus 1,300 samples. From the total of samples, 910 were used to obtain the describing models and 390 were used for testing. In other words, from the 50 samples of each character, 35 were used for model construction and 15 for testing. Figure 3 shows one sample of characters: "a", "n", "m" and "o".

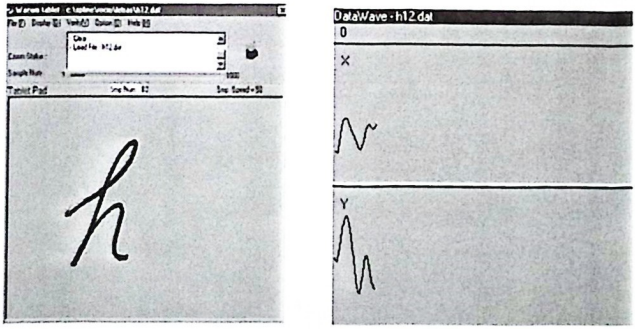


Fig. 2. Captured data from the trace of character "h". (a) Trace. (b) Obtained signals in the x and y axes.

2.3 Signal Approximation

When the signal of the character is acquired, it is accompanied with some high frequency noise produced by the small vibrations introduced by the movement of the hand. To reduce this kind of noise a Daubechies 1 wavelet was applied. Only the approximation part was taken. Figure 4 (a) shows the original trace of a sample of character "e" and the one obtained (Figure 4 (b)) by wavelet processing the character as explained.

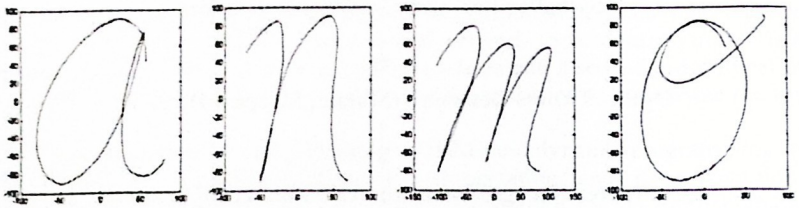


Fig. 3. A sample of the trace of characters: "a", "n", "m" and "o".

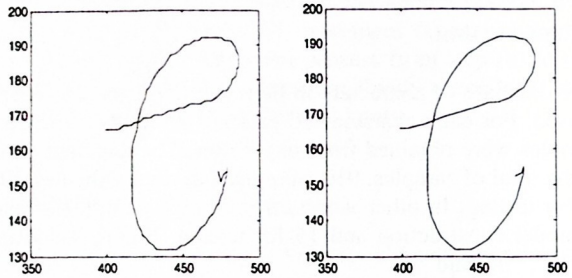


Fig. 4. (a) Original trace of character "e" and (b) its corresponding trace after wavelet approximation.

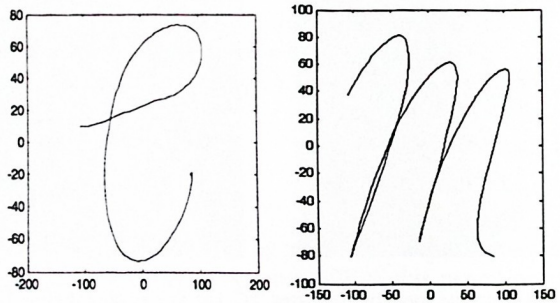


Fig. 5. Traces of letters "e" and "m" after filtering and normalization.

2.4 Post-Filtering and Normalization

To yet reduce high frequency components introduced by the vibration of the hand, a low pass Butterworth filter of order 5 was applied to the character traces in both the x and the y axes. After low pass filtering the character, it is normalized in position, size and time. Normalization in position and size is performed by an affine transformation of the character. Normalization in time is done by interpolation and decimation. Figure 5 shows the traces of the characters "e" and "m" after filtering normalization.

2.5 Feature Extraction

Feature extraction, in the content of this work consists on obtaining the optimal nodes from the signals in the x and y axes of the traced characters. To accomplish this, the SLALOM method well described in [5] was used. The general schema of the application of the SLALOM method to obtain the optimal points of the characters trace is shown in Fig. 6.

SLALOM METHOD

The Slalom method is one type of natural Spline function which must satisfy the following two conditions:

1. The difference between the Spline function $g(x)$ and a given f_i (i=1...M) must be smaller than a previously determined value, δ.

$$|g(x) - f_i| \leq \delta \qquad i=1,2,....,M \tag{1}$$

2. The Spline function g(x) must be a smooth function, that does not need to cross over for every given points f_i (i=1...M).

Figure 5 shows the smooth Spline function g(x) generated by Slalom method and sampling points f_i (i=1...M).

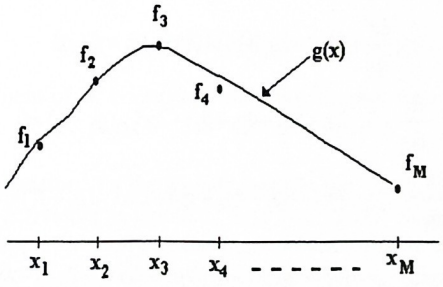


Fig. 6. Smooth function.

To satisfy the two conditions given above, an error function J[g] must be minimized. We can define the first derivative J'[g] of the error function J[g] as Eq. (2).

$$J'[g]=\int\left(\frac{d^2}{dx^2}g(x)\right)^2 dx+\alpha\sum_{i=1}^{M}(g(x_i-f_i)^2 dx \tag{2}$$

and the first and second derivative of g(x) correspondent to i+1th knot in discrete way, which is written as:

$$\dot{g}_{i+1}=\frac{g_{i+1}-g_i}{x_{i+1}-x_i}=\frac{g_{i+1}-g_i}{\Delta} \tag{3}$$

$$\dot{g}_{i+1}=\frac{1}{\Delta}\cdot\left(\frac{g_{i+1}-g_i}{\Delta}\frac{g_i-g_{i-1}}{\Delta}\right)=\frac{g_{i+1}-2g_i+g_{i-1}}{\Delta^2} \tag{4}$$

where Δ is the interval between the i-th and the i+1-th knots. Supposing that intervals between two consecutive knots is equal to 1, we have

$$\dot{g}_{i+1}=g_{i+1}-2g_i+g_{i-1} \tag{5}$$

Next, Eq. (2) can be rewrite using as,

$$J'[g]=\sum_{j=2}^{N-1}\left(g_{j+1}-2g_j+g_{j-1}\right)^2+\alpha\sum_{i=1}^{M}\left(g_{j_i}-f_i\right)^2 \tag{6}$$

where N is the samples number and M is the number of knots, g_i is the i-th value from Spline function g(.) and g_{ji} is the equivalent value of g(.) of the i-th knot.

The minimization problem of J'[g] can be solve as follows.

$$\frac{\partial J'}{\partial g_k}=0, \qquad k=1...N \tag{7}$$

By substituting Eq. (6) into Eq. (7), we can get,

$$\frac{\partial J'}{\partial g_1}=2(g_1-2g_2+g_3)+2\alpha\delta_{1,\Omega}(g_1-f_1)=0$$

$$\frac{\partial J'}{\partial g_2}=2(-2g_1+5g_2-4g_3+g_4)+2\alpha\delta_{2,\Omega}(g_2-f_2)=0$$

$$\frac{\partial J'}{\partial g_3}=2(g_1-4g_2+6g_3-4g_4+g_5)+2\alpha\delta_{3,\Omega}(g_3-f_3)=0$$

$$\frac{\partial J'}{\partial g_{N-2}}=2(g_{N-4}-4g_{N-3}+6g_{N-2}-4g_{N-1}+g_N)+2\alpha\delta_{N-2,\Omega}(g_{N-2}-f_{N-2})=0$$

$$\frac{\partial J'}{\partial g_{N-1}}=2(-2g_{N-3}+5g_{N-2}-4g_{N-1}+g_N)+2\alpha\delta_{N-1,\Omega}(g_{N-1}-f_{N-1})=0$$

$$\frac{\partial J'}{\partial g_N} = 2(g_{N-2} - 2g_{N-1} + g_N) + 2\alpha\delta_{N,\Omega}(g_N - f_N) = 0 \tag{8}$$

to get the g_k, for $k = 1, 2, \ldots, N$, the following lineal equations must be resolved

$$\begin{bmatrix} 1 & -2 & 1 & & & & & \\ -2 & 5 & -4 & 1 & & & & \\ 1 & -4 & 6 & -4 & 1 & & & \\ & \ddots & \ddots & \ddots & \ddots & & & \\ & & 1 & -4 & 6 & -4 & 1 & \\ & & & 1 & -4 & 5 & -2 & \\ & & & & 1 & -2 & 1 & \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ \vdots \\ g_{N-2} \\ g_{N-1} \\ g_N \end{bmatrix} = \begin{bmatrix} \alpha\delta_{1,\Omega}(f_1 - g_1) \\ \alpha\delta_{2,\Omega}(f_2 - g_2) \\ \alpha\delta_{3,\Omega}(f_3 - g_3) \\ \vdots \\ \alpha\delta_{N-2,\Omega}(f_{N-2} - g_{N-2}) \\ \alpha\delta_{N-1,\Omega}(f_{N-1} - g_{N-1}) \\ \alpha\delta_{N,\Omega}(f_N - g_N) \end{bmatrix} \tag{9}$$

where δ is the sampling space and $\delta_{j,\Omega}$ satisfies,

$$\delta_{j,\Omega} = \begin{cases} 0 & j \notin \Omega \\ 1 & j \in \Omega \end{cases} \tag{10}$$

$\delta_{j,\Omega} = 0$, when j is different from any knot position and $\delta_{j,\Omega} = 1$, when j corresponds with the some knot position.

After the Slalom method is applied and optimal knots are obtained, the error E, can be obtained by Eq. (11).

$$E = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - g(x_i))^2 \tag{11}$$

To minimize the error E, the steepest descent method is applied. The steepest descent method is the simplest gradient methods used to minimize a given error function. Then the optimal knot's position update is given by Eq. (12).

$$x_{k+1} = x_k - \lambda_k \nabla f(x_k) = x_k - \lambda_k g(x_k) \tag{12}$$

After the adjustment of all knots position is performed, we analyze the distance of all consecutive knots.

Optimal node initialization. To the signals, both in the *x* and *y* axes, of the trace of a character a second derivative is applied. The use of the second derivative allows determining the change of velocity at the moment the trace is done. However at it is known the application of the second derivative amplifies the small changes in the signal, giving as a result a noisy signal. To reduce the influence of this introduced noise, the derived signals (in both axes) are smoothed again by means of a low pass filter. Figure 7 shows the processed signals of letter "m" and the second derivative of these signals, respectively. The derived signals are divided into segments according to

the number of zero crossings. In this case, the value of "0" in the second derivative means that the change of velocity is zero. For each segment, the data with maximal absolute value are considered as the initial nodes. Figure 8 shows the initial nodes of letters "m" and "e". These nodes are not optimal due to errors between signals are too big. To reduce the magnitude of this error we take the initial nodes f_i and we get a smooth and continuous function $g(t)$ by means of SLALOM method [5].

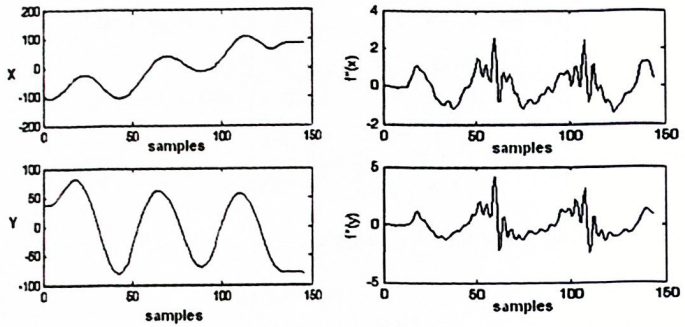


Fig. 7. (a) Signals corresponding to letter "m" after preprocessing and (b) low pass filtered of the second derivative of the same letter.

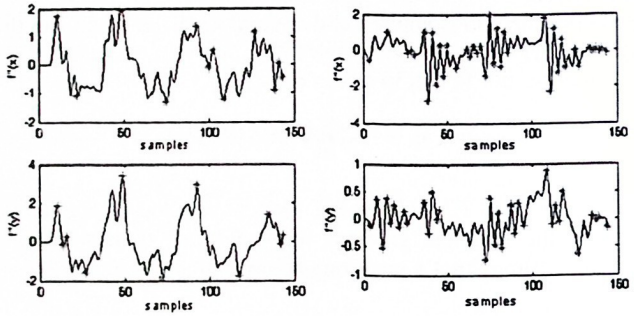


Fig. 8. (a) Initial nodes of letter "m" and (b) initial nodes of letter "e".

Obtaining the optimal nodes. By applying again SLALOM method we obtain the intermediate nodes generated by a smooth and continuous function. Figure 9 shows the nodes obtained by taking the first term of the SLALOM method along with the reconstructed signal. As we can observe from Fig. 9, the obtained points are redundant due to each node represent the same position. By eliminating these redundant nodes and by applying the step and decent method to include the second term of equation (9) we can thus obtain the optimal set of nodes.

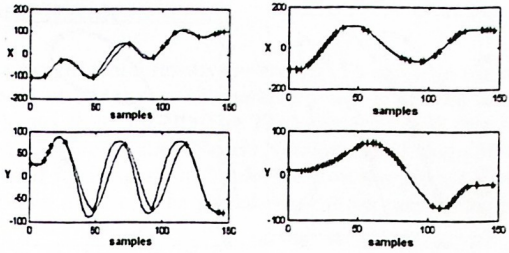


Fig. 9. Original signals (line "-") and reconstructed signals (line "-.") from the obtained nodes ("*") by means of the SLALOM method [5]. (a) letter "m" and (b) letter "e".

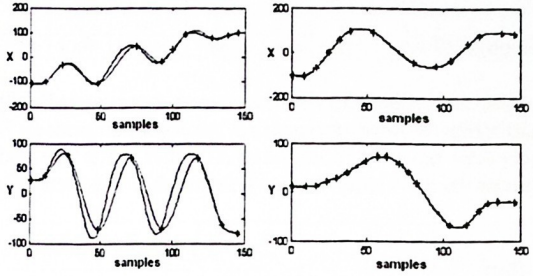


Fig. 10. Original signals (line "-") and reconstructed signals (line "-.") from the obtained nodes ("*"), (a) letter "m" and (b) letter "e".

Optimal node adjustment. The adjustment of the number and positioning of the optimal nodes is performed by using again the step and descent method. Fig. 10 shows the results of the elimination of the redundant nodes, while Fig. 11 shows the results of the adjustment. These operations are applied iteratively while arrives to desired number of nodes (20). Figure 12 shows the optimal nodes (both in the x and y axes), the original trace and the reconstructed trace from this set of nodes.

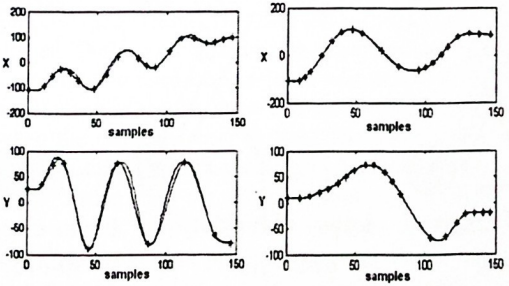


Fig. 11. Adjusted nodes by step and descent method. Original signals (line "-") and reconstructed signals (line "-.") from the obtained nodes ("*"), (a) letter "m" and (b) letter "e".

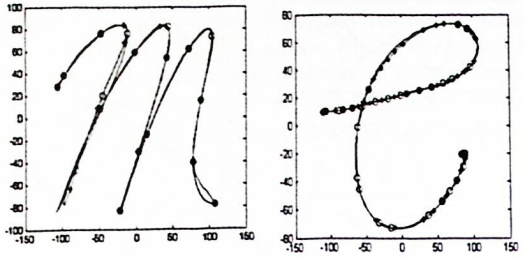


Fig. 12. Original traces and reconstructed traces from the optimal nodes. "*"means optimal node in the *x* axe, and "o" means optimal node in the *y* axe.

2.6 Character Recognition

As mentioned before, the proposed system uses one three-layer Backpropagation type neuronal networks (Fig 13) for processing the optimal nodes. For this features processing the networks have 40 input data (20 nodes for x axe and 20 nodes for y axe), with 40 neurons in the hidden layer. This number was determined after several test.

Learning Algorithm

The learning algorithm used for updating the system coefficient matrix is the very well known backpropagation algorithm.

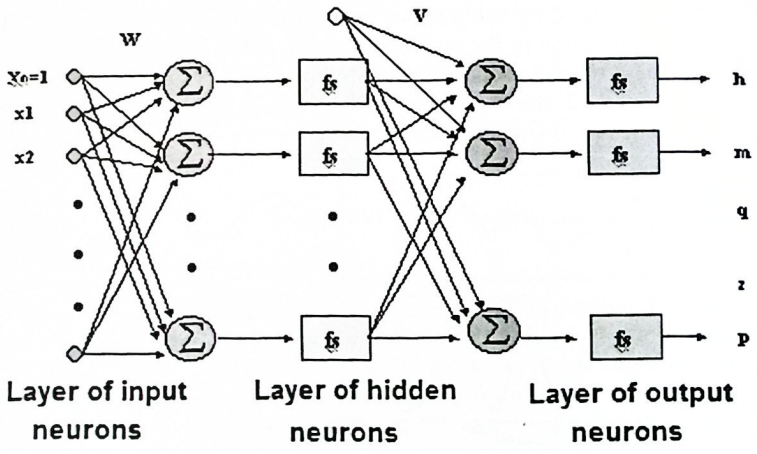


Fig. 13. The structure of the neural network used to recognize the characters.

3 Experimental Results

In this section the experimental results are shown. To evaluate the proposed system, a data base composed of 3,900 cursive characters was generated consist of 50 cursive characters of 26 letters from 3 users. The 2730 characters (35 characters of 26 letters) were used to train Backpropagation Neural Network 1170 characters (15 characters of 26 letters) were used for evaluation. Table 1 shows the proposed system recognition rate using the training data set. The global recognition rate of the proposed system is 99.81%.

Table 1. Recognition rate with characters used during training.

a	b	c	d	e	f	g	h	i	j	k	l	m
100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	99.04%	100%	99.04%
n	o	p	q	r	s	t	u	v	w	x	y	z
100%	99.04%	98.09%	100%	99.04%	100%	100%	100%	100%	100%	100%	100%	100%

Table 2 shows the recognition rate was carried out with characters not used during training. The overall recognition rate of the proposed system is 97.8%.

Table 2. Recognition rate with characters not used during training.

a	b	c	d	e	f	g	h	i	j	k	l	m
95.5%	95.5%	100%	97.7%	100%	100%	100%	93.3%	100%	100%	93.3%	100%	95.5%
n	o	p	q	r	s	t	u	v	w	x	y	z
97.7%	95.5%	95.5%	100%	93.3%	100%	95.5%	100%	100%	100%	97.7%	100%	97.7%

Classification percentages for a similar methodologies described in the literature are from 85% and 98% [1]-[4]. Comparing the proposed system with others similar systems has a similar recognition rate for training and testing data set..

4 Conclusions

In this paper a new methodology for the recognition of cursive manuscript characters has been presented. The SLALOM method was used to obtain the optimal knots of each character. These optimal knots are considered as the describing features of each character, which were used like an input vector in a Backpropagation Neural Network. Computer evaluation results show that the proposed system provides a good recognition rate when the same database is used for training and testing, as well as when both databases are different, obtaining a 99.34 % recognition rate for training data set and 97.43 % testing data set. These results can be considered quite good thinking that the characters recognized are cursive and have a certain shape deformation degree. The recognition percentage using the proposed system is good enough against the percentages obtained with similar proposals systems described in the literature where the results are around 85% to 98% recognition rate.

Acknowledgements. This research was supported by CONACYT and SIP under grants 46805 and 20071438, respectively.

References

1. R. Plamondon, S. N. Srihari, "On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey", IEEE Trans. on Pattern Analysis and Machine Intell., vol. 22, no. 1, 2000, pp. 63-84.
2. S. Manke y U. Bodenhausen, "A Connectionist Recognizer for On-line Cursive Handwriting Recognition", Proceedings of the Int. Conf. Acoustic, Speech and Signal Processing, ICASSP '94, vol. 2, 1994, pp.633-636.
3. R. Plamondon y F. J. Maarse, "An Evaluation of Motor Models of Handwriting", IEEE Trans. System, Man, and Cybernetics, vol.19, no. 5, 1989, pp. 1060-1072.
4. B. Sin, J. Ha, S. Oh y J. Kim "Network-Based Approach to Online Cursive Script Recognition", IEEE Trans. System, Man, and Cybernetics, vol. 29, no. 2, 1999, pp. 321-328.
5. Y. Isomichi, "Inverse-Quantization Method for Digital Signals and Images", IEICE Transactions, Vol. J64-A, no. 4, pp. 285-292, 1981.
6. T. Huang y M. Yasuhara, "A Total Stroke SLALOM Method for Searching for the Optimal Drawing Order of Off-line Handwriting", Proc. IEEE Int. Conf. on System, Man and Cybernetics, vol. 3, 1995, pp. 2789-2794.
7. Hailong Liu; Xiaoqing Ding, "Handwritten character recognition using gradient feature and quadratic classifier with multiple discrimination schemes", Proceedings Eighth International Conference on Document Analysis and Recognition, Vol. 1 pp 19 – 23, 2005.
8. Teng Long; Lian-Wen Jin; Li-Xin Zhen; Jian-Cheng Huang "One stroke cursive character recognition using combination of directional and positional features", IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 5, pp. 449-452, 2005.

Cluster Tree Self Organizing Map for Developing Image Retrieval System

Hamed Shahbazi[1], Mohsen Soryani [2], Nasser Mozayani [2], and Mahmood Fathy [2]

Computer Engineering Department, Iran University of Science and Technology
Narmak, Tehran 16844, Iran
[1] hshahbazi@comp.iust.ac.ir
[2] {soryani, mozayani, mahfathy}@iust.ac.ir

(Paper received on June 18, 2007, accepted on September 1, 2007)

Abstract. In this paper, a novel structure of hierarchical self organizing map called Cluster Tree Self Organizing Map (CT-SOM) is proposed. This structure is a hierarchical representation of the cluster of a data set that will be used for image retrieval. Neural units of CT-SOM are given multi labels and each label hints at one image in database. For each different image's feature, a CT-SOM is formed. These representations include color, texture and shape. Using evidence accumulation we have facilitated automatic combination of responses from multiple CT-SOMs and their hierarchical levels. A new relevance feedback technique is also used based on user's preferences for finding image resemblance in each category. We have performed experiments and tested the proposed approach on an image database constructed from Corel photo gallery.

1 Introduction

In recent years Content-Based Image Retrieval (CBIR) has been a subject of very extensive research field and many projects have been started to research and develop efficient CBIR systems. Despite some breakthroughs made in the field, it is generally understood that the problem is still far from being solved. Some of the popular CBIR systems include QBIC project [1], MIT's Photobook [2], VisualSeek [3], PicSOM [4] and lot more.

Heretofore, variant structure of hierarchical self-organizing map is used to develop indexing structure. For instance tree structured SOM which are introduced in PicSOM, and four-level R-tree SOM [5] which are used for image retrieval. The mentioned indexing structures have a problem of having a large overlapping area among nodes, causing the retrieval process to inspect a large number of image items. Also until relevance feedback modifies the structures remarkably, the approach for combination of the structures' results is weak.

A cluster tree [6] is a hierarchical representation of the cluster of a data set. This index structure organizes the data based on their different level of clustering information from coarse to fine. Here, we develop structures of SOM which represent a cluster tree of data and decrease overlapping area among nodes. These structures are called Cluster Tree Self Organizing Map (CT-SOM). For each visual feature, one CT-

© H. Sossa and R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 81-90

SOM is developed. Partition of clusters is organized from precise bottom level to coarse top level of CT-SOMs. Following production of partitions in CT-SOMs' levels, evidence accumulation [7] is used to facilitate automatic combination of responses from multiple hierarchical structures. A new relevance feedback technique is also used based on user's preferences for finding image resemblance in each category.

The remainder of this paper is organized as follows. Section 2 describes cluster tree self organizing map. Relevance feedback to refine query is proposed in section 3. Visual content features, is given in section 4 and section 5 gives the performance and experimental results.

2 Cluster Tree Self Organizing Map

The SOM [8] carries out vector quantization and multi-dimensional scaling at the same time. In step index $t = 0,\ 1,\ldots\ t_{\max} - 1$, an input vector $X(t)=[X_1(t),..X_m(t)]^T$ is presented to the network and unit $i(X)$ with synaptic weight vector $W_j(t)=[W_{j1}(t),..W_{jm}(t)]^T$ is selected as the Best Match Unit (BMU), based on the best matching criterion (1).

$$i(X) = \arg\min_j \| X(t) - W_j(t) \|,\quad j = 1, 2, \ldots, l \tag{1}$$

Where l is the number of units and $\| \|$ represents Euclidian distance. Consequently, the weight vectors are updated according to (2).

$$W_j(t+1) = W_j(t) + \eta(t) h_{j,i(x)}(t)(X(t) - W_j(t)) \tag{2}$$

Where $h_{j,i(x)}(t)$ is topological neighborhood function which the typical choice of it is Gaussian function (3).

$$h_{j,i(x)}(t) = \exp(-\frac{d_{j,i}^2}{2\sigma^2(t)}) \tag{3}$$

Where $d_{j,i}^2 = \| r_j - r_i \|^2$ and r_j defines the position of excited neuron j, and r_i defines the discrete position of winning neuron i. In order to construct the CT-SOM, below steps are done:

Step1) The size of first level is $r_1 * c_1$ and initial standard deviation of neighborhood function is σ_0. This level, after training is fixed and each neural unit on it is given labels from the database image nearest to it. In other words CT-SOM's neural units are given multi labels and each label hints at one image in database. We called the neural unit and its labels as Unit Cluster (UC).

Step2) The map is divided into squares with size $\Lambda \times \Lambda$ (Λ is odd) and the k-mean algorithm is performed on map as follow:

- Begin with $K = (r_1 \times c_1)/(\Lambda \times \Lambda)$ clusters which their centroids are initialized with the centers of mentioned squares. The cluster prototype matrix is shown with $W = [w_1, ..., w_K]$.
- Assign each unit of map to the nearest cluster C_l .i.e.

$$U_j \in C_l, if \left\| U_j - w_l \right\| < \left\| U_j - w_i \right\| \quad (4)$$
$$for \ \ j = 1, ..., (r_1 \times c_1), i = 1, ..., K \ \ and \ \ i \neq l.$$

- Recalculate the cluster prototype matrix based on the current partition.
- Repeat steps 2 and 3 until there is no change for each cluster.

We called these clusters as Map Cluster (MC) which consists of some UCs. Fig.1 shows two MCs for Λ=3 .

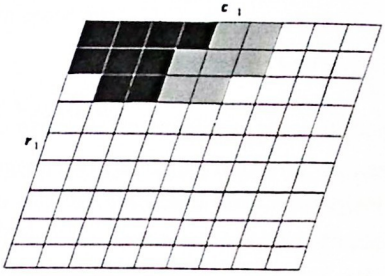


Fig. 1. Two Map Clusters (MCs) are shown. Each MC contains some Unit Clusters (UCs)

Step3) Every later level is developed on previous level which its map's size is number of MCs on previous level. This means that every unit on later level represents one MC on previous level.

Each later level, after construction is trained and its UCs and MCs are determined. In training of later *jth* level, the BMU is determined in this way that, the input object is presented to the first level and the BMU and therefore the MC that this unit belongs to it is determined. Then the BMU of the next level is unit that represents the determined MC on previous level. This iteration goes on until reach *jth* level.

Step3 is iterated until efficient partitions with desirable number of clusters (size of last level) are gained. In the end of this step, cluster tree is produced which is a hierarchical representations of the cluster of a data set. This hierarchical structure organizes the data based on their different level of clustering information from coarse to fine, providing an index structure of data. For each visual feature we construct a CT-SOM separately. Since various feature classes are not necessarily linearly-related so we consider each map as a partition of UCs and use evidence accumulation to combine them. Evidence accumulation gives us desired co-association matrix that indicates the degree of resemblance between images.

Assume that N is the number of database images, B is the number of partitions and the final partitions of UCs which acquired from CT-SOMs' hierarchical levels are $\mathrm{P} = \{P^1, P^2, ...P^B\}$. Since processing of N^*N proximity matrix has computational complexity for large value of N, so we pre-compute a N^*q matrix which stores the indices of the q nearest neighbors for each of the N images. In our experiment, we set q equal to 15. The nearest-neighbor matrix can be computed as a preprocessing step by using branch and bound algorithm [9].
Co-association matrix is calculated as follow:

1. C = co-association matrix with dimension N^*q is initialized to a null matrix.
2. For each data partition $p^l \in \mathrm{P}$, do:
   2.1. Update the co-association matrix as:
   For each image pair (i, j) in the qth neighbor list which belongs to the kth UC in p^l, set:

$$C(i,j) = C(i,j) + \frac{\gamma_{l,k}}{B} \tag{5}$$

Where $\gamma_{l,k} \in [0,1]$ and its value is minimum for coarse clusters (top levels) and maximum for precise one (low levels). It means that two images in precise clusters are more similar than two images in coarse one. Each UC has its $\gamma_{l,k}$ value and gives special amount of similarity between its members. We use relevance feedback mechanism to modify the value of $\gamma_{l,k}$ and improve queries result. $\gamma_{l,k}$ may be dependent to type of visual features.

In actual implementation to retrieve image, CT-SOMs are browsed from top to bottom and in each level, the BMU and consequently the UC which is bound to it, are determined. The labels of this UC together with 10 labels from co-association matrix with highest resemblance value to prior selected labels are selected. In browsing to lower levels, the search space for BMU is restricted to the MC which BMU on top previous level represents it.

Finally for each CT-SOM, one set of labels is acquired which is shown with $S = \{S_1, S_2, ..., S_\Delta\}$. Where Δ is the number of CT-SOMs. Images that are presented to user, to get his/her preferences, are intersection between these sets:

$$R = \bigcap_{i=1}^{\Delta} C_i \tag{6}$$

3 Refining Queries Using Relevance Feedback

The System tries to learn the user's preferences from the interaction with him/her and then satisfies own responses accordingly. We use a relevance feedback approach in which the results of multiple CT-SOMs are combined automatically by using the implicit information from the user's responses during the query session. This can be implemented simply by marking each kth UC on lth map with $\gamma_{l,k} \in [0,1]$ as similarity weight. Initial values of $\gamma_{l,k}$ is minimum for coarse clusters (top levels) and maximum for precise (low levels).

The user's preferences for each images is either relevant or irrelevant. For each image pair (i, j) in the images which are shown to user, if they are relevant then the $\gamma_{l,k}$ of UCs which this pair are belong to it, is increased and vice versa. As seen in (5), the modification of $\gamma_{l,k}$ affects the value of co-association matrix which will be recomputed after some interactions.

4 Visual Content Features

Feature selection is not restricted and every feature or description of it can be added. In our experiment fuzzy color histogram, entropy and shape histogram are selected which are described in later subsection.

4.1 Fuzzy Color Histogram

The Fuzzy Color Histogram (FCH) [10] of image I can be expressed as $F(I) = [f_1, f_2, \ldots f_n]$, where

$$f_i = \sum_{j=1}^{N} \mu_{ij} P_j = \frac{1}{N} \sum_{j=1}^{N} \mu_{ij} \tag{7}$$

μ_{ij} is the membership value of the jth pixel in the ith color bin and p_j is the probability of jth pixel selected from image I. Let M (8) is the membership matrix and m_{ij} is the membership value of the jth fine color bin distributing to the ith coarse color bin:

$$M = [m_{ij}]_{n^* n}. \tag{8}$$

The FCH of an image can be directly computed as follows:

$$F_{n^{*}1} = M_{n^{*}n'} H_{n'^{*}1} \tag{9}$$

Where membership matrix M is pre-computed only once and can be used to generate FCH for each database image. M is computed as follow:

- Fine uniform quantization in RGB color space is performed by mapping all pixel colors to n' histogram bins. Then, the n' colors are transformed from RGB to CIELAB color space.
- Using FCM clustering technique [11], these colors in CIELAB color space is classified to clusters, which each cluster representing an FCH bin.

The FCM minimizes an objective function J_m, which is the weighted sum of squared errors within each group, and is defined as follows:

$$J_m(U,V;X) = \sum_{k=1}^{n}\sum_{j=1}^{c} u_{ik}^{m} \left\| x_k - v_i \right\|_A^2 \tag{10}$$

$$1 < m < \infty$$

Where, $V = [v_1, v_2, ...v_c]^T$ is a vector of unknown cluster prototypes. The value of u_{ik} represents the membership of the data Point x_k from the set $X = \{x_1, x_2, ...x_n\}$ with respect to the ith cluster. The inner product defined by a norm matrix A defines a measurement of similarity between a data point and the cluster prototypes, respectively. The fuzzy clustering result of FCM algorithm is represented by matrix $U = [u_{ik}]_{n^{*}n'}$. u_{ik} is referred to as the grade of membership of color x_k with respect to cluster center v_i. Thus, the obtained matrix $U_{n^{*}n'}$ can be viewed as the desired membership matrix $M_{n^{*}n'}$ for computing FCH, i.e. $M_{n^{*}n'} = U_{n^{*}n'}$. Moreover, the weighting exponent m in FCM algorithm controls the extent of membership shared among the fuzzy clusters.

4.2 Entropy Histogram

The co-occurrence matrix [12] is a two-dimensional histogram which estimates the pair-wise statistics of gray level. The $(i, j)th$ element of the co-occurrence matrix represents the estimated probability that gray level i co-occurs with gray level j at a specified displacement d and angle θ. Entropy histogram is acquired as follow:

- Conversion of color image to gray image.
- Dividing image into 2*2, 4*4, 8*8, and 16*16 rectangular regions as in color case.

- Obtaining co-occurrence matrix of four (horizontal 0^{o}, vertical 90^{o} and two diagonal $45^{o}, 135^{o}$) orientation in region and normalize entries of four matrixes to [0, 1], by dividing each entry by total number of pixels.
- Extracting average entropy value from four matrixes.

$$e = \frac{-\sum_k \sum_i \sum_j p(i,j)\log(p(i,j))}{4}, k = 1,2,3,4 \tag{11}$$

- Constructing entropy histogram of regions' entropy.

4.3 Shape Histogram

This feature describes the distribution of edge directions in various parts of the image and thus reveals the shape in a low-level statistical manner [13]. It is calculated in five separate zones of the image. The first zone is formed by extracting from the center of the image, a circular zone whose size is approximately one-fifth of the area of the image. Then the remaining area is divided into four zones with two diagonal lines. Shape histogram feature is based on the histogram of the eight quantized directions of edges in the image. When the histograms are separately formed in the same five zones, as before, an 8* 5 = 40 dimensional feature vector is obtained.

5 Performance and Experimental Result

Corel gallery product [14] contains 59995 photographs and artificial images with a very wide variety of subjects. Image collection which we used in our experiment is a set from the Corel Gallery. First, we consider 48 semantic concepts (Classes) and then select 12000 images from Corel gallery and give 48 membership values to each of them. Each membership value determines the belonging degree of image to one concept. The evaluation of performance for retrieval system can be mathematically formulated as follow: Suppose that the size of database is N and the number of semantic concept is C. Membership value of ith images to semantic concept can be expressed by $H_i = \{h_{i1}, h_{i2}, ..., h_{ic}\}$, and for all images in database it is expressed by $M = \{H_1, H_2, ..., H_N\}$. Let in time t, Queries on image with membership $Q_t = \{q_{t1}, q_{t2}, ..., q_{tc}\}$ is given to system and $K(t)$ images are retrieved as a query result. Recall is defined as:

$$\text{Re}call = \frac{Number\ of\ images\ retrieved\ and\ relevant}{Total\ number\ of\ relevant\ images\ in\ the\ database} \tag{12}$$

Recall is expressed with $R(t)$ for time t, and Precision is defined as:

$$\Pr ecision = \frac{Number\ of\ images\ retrieved\ and\ relevant}{Total\ number\ of\ retrieved\ images} \tag{13}$$

$P(t)$ is used to determine Precision for time t. We have chosen to show the evolution of precision as a function of Recall. Using above assumption, $R(t)$ and $P(t)$ is defined as follow:

$$R(t) = \frac{\sum_{i=1}^{K(t)} \frac{1}{1 + \sum_{j=1}^{c} (\|q_{ij} - h_{ij}\|)^2}}{\sum_{i=1}^{N} \frac{1}{1 + \sum_{j=1}^{c} (\|q_{ij} - h_{ij}\|)^2}} \tag{14}$$

$$P(t) = \frac{\sum_{i=1}^{K(t)} \frac{1}{1 + \sum_{j=1}^{c} (\|q_{ij} - h_{ij}\|)^2}}{K(t)} \tag{15}$$

The intermediate values of $P(t)$, first, display the initial accuracy of the system and then, show how RF mechanism is able to adapt the class. For our experiment, average precision, as a function of average recall is changed as indicate in Fig 2.

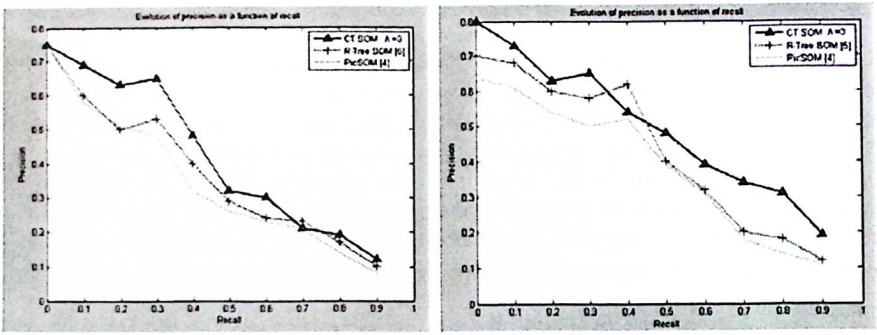


Fig. 2. The evolution of precision as a function of recall. (a) Building query. (b) Tiger query.

Some of Exemplar Queries that is given to the system are shown in Fig. 3 and 4.

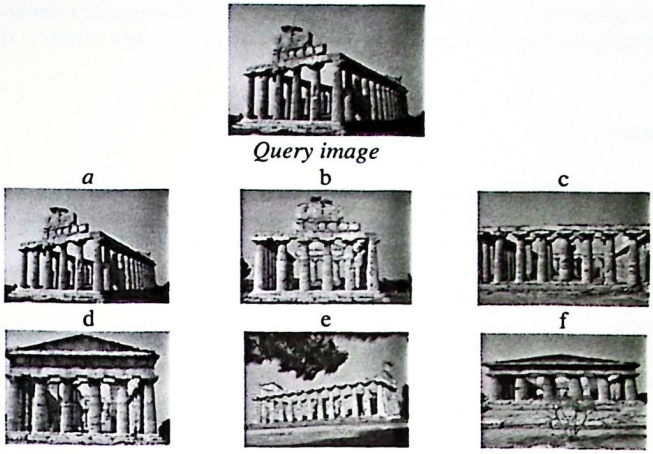


Fig. 3. Retrieval result for query of image with Building semantic.

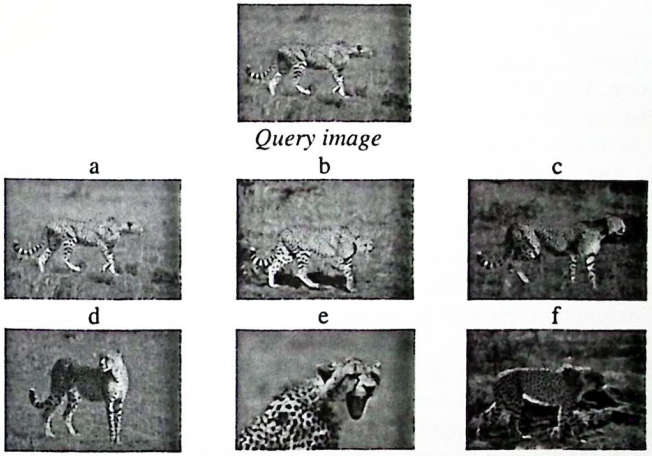


Fig. 4. Retrieval result for query of image with Tigers semantic.

6 Conclusion

The CT-SOM that is presented in this paper is very useful for large image data sets. This system has three advantages: First, it used the SOM with multi labeled neural units and thus organizes images into a hierarchical structure without overlapping area between nodes. Second, hierarchical maps are from coarse top level to precise bottom

level so the query time is largely reduced. Third the mechanism of relevance feedback used in the proposed system improves the performance of image retrieval drastically.

References

1. Flickner, M., Sawhney, H., Niblack, W., Ashley, J., Huang, Q., Dom, B., Gorkani, M., Hafner, J., Lee, D., Pettovic, D., Steele, D., and Anker, P. "Querying by image and video content: The QBIC system," *IEEE Trans. Computers 25*, 1995, pp.23–32.
2. Pentland, A., Picard, R.W., and Sclaroff, S. "Photobook: tools for content-based manipulation of image databases. In: Storage and Retrieval for Image and Video Databases," II. In: *SPIE Proceedings Series*, Vol. 2185. San Jose, CA, USA, 1994.
3. Smith, J. R., and Chang, S. F. "VisualSeek: a fully automated content-based image query system," *Proc. ACM Multimedia*, 1996, pp. 87–98.
4. Laaksonen, J., Koskela, M., Laakso, S., and Oja, E. "PicSOM: content-based image retrieval with self-organizing maps," *Elsevier, Pattern Recog.* Lett.21, 2000, pp. 1199–1207.
5. Subramanyam Rallabandi, V. P., Sett, S.K. "Image retrieval system using R-tree self-organizing map," *Elsevier, Data & Knowledge Engineering*, 2006.
6. Dantong, Y., and Zang, A. "Cluster Tree: Integration of cluster representation and nearest neighbor search for large data base in high dimensions," *IEEE Transaction on knowledge and Data Eng*, Vol.15, No.5, 2003, pp.1316-1337.
7. Fred, Ana L.N., Jain, and Anil K. "Combining Multiple Clusterings Using Evidence Accumulation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, No.6, 2005.
8. Kohonen, T. *"Self-Organizing Maps,"* third ed., Springer, New York, 2001.
9. Kamgar-Parsi, B., and Kanal, L.N. "An Improved Branch and Bound Algorithm for Computing k-Nearest Neighbors. *Pattern Recognition ," Letters*, vol. 1, 1985, pp.195-205.
10. Ju, Han., and Kai-Kuang, Ma. "Fuzzy Color Histogram and Its Use in Color Image Retrieval," *IEEE Trans. Image Processing*, Vol. 11, No. 8, 2002.
11. Rezaee, M. R., LeLieveldt, B. P. F., and Reiber, J. H. C. "new cluster validity index for the fuzzy c-means," *Pattern Recognition,* vol. 19, 1998, pp.237–246.
12. Haralick, R. M., Shanmugam, K., and Dinstein, I. "Texture features for image classification," *IEEE Transactions on Systems, Man, and Cybernetics, SMC-3(6)*, 1973, pp.610–621.
13. Brandt, S., and Laaksonen, J. E. Oja. "Statistical shape features in content-based image retrieval," *In: Proceedings of 15th International Conference on Pattern Recognition*, Barcelona, Spain, Vol. 2, 2000, pp.1066±1069.
14. Gunther, N.J., Beretta, and G. "A benchmark for image retrieval using distributed system over the internet," BIRDS-I HP Labs, 2000 Available from: <www.hpl.hp.com/techreports/2000/HPL-2000-162.html>.

Feature Extraction and Dimensionality Reduction

Using Testor Theory to Reduce the Dimension of Neural Network Models

Roberto A. Vázquez [1] and Salvador Godoy-Calderón [2]

Computing Research Center CIC-IPN,
Av. Juan de Dios Bátiz, esquina con Miguel Othón de Mendizábal,
Mexico City, 07738, MEXICO.
Contact: [1]ravem@ipn.mx, [2]sgodoyc@cic.ipn.mx

(Paper received on July 10, 2007, accepted on September 1, 2007)

Abstract. Most of the Neural Network models proposed during the last few years are capable of solving several complex problems such as recognition, forecast or reconstruction of different phenomena. A crucial feature of these models is that they tackle a wide variety of classification problems, and although these models work accurately within a limited particular context, they require numerous resources; this makes its efficient hardware implementation nearly impossible. In this paper we propose a novel alternative which uses Testor Theory, which is a useful tool in pattern recognition within the logical-combinatorial approach, to reduce the dimension of neural network models. We test the accuracy of the proposed method by recognizing real-world objects in images. We show that, under some circumstances, it is possible to reduce the dimension of a neural network without affecting its capability to solve classification problems or modifying its effectiveness.

1 Introduction

Humans have several complex cognitive capabilities such as memorizing, recalling, learning and recognizing. In the last 50 years, scientists and researchers of different communities have strived to implement these capabilities into a computer. Along these years, several approaches for achieving that goal have emerged, for example neural networks. Since the rebirth of neural networks, several models inspired in neurobiological processes have been proposed. Such models are often dedicated and incorporate some existing clustering or classification algorithm. Among these models, perhaps the most popular one is the feed-forward multi-layer perceptron trained with the back-propagation algorithm [10].

Other very popular neural models are associative memories. Several of these associative models have been proposed, for example: Anderson [6] presents a simple neural network generating an interactive memory; Kohonen [7] presents an unsupervised learning network as an explanation of the existence of ordered maps in the brain. Other associative models can be found in [3], [9], [26], [27], [28], [29], [32] and [33]. Advantages of neural networks are: adaptability, robustness, and ease of implementation in software.

Most of the neural network models are capable of solving several kinds of complex problems such as face recognition, object recognition, plate recognition, hand-writing

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 93-103

recognition and some other classification problems. Despite the fact that these models work accurately, they require too many resources, thus making their efficient implementation on hardware nearly impossible.

In this paper we propose a novel alternative to reduce the dimension of neural network models by using Testor Theory, which is a useful tool in pattern recognition under the logical-combinatorial approach. We test the accuracy of the proposal by solving an object recognition problem using several images of realistic objects. In the experiments, we expect to reduce the dimension of the neural network models used for solving the problem by reducing the dimensionality of the features they take as input.

2 A Survey of Dimensional Reduction Techniques

When working with high-dimensional datasets it is often the case that not all the measured variables have the same "relevance" for understanding the underlying phenomena of interest. Certain computationally-expensive methods can construct predictive models with high accuracy from high-dimensional data. Still reducing the dimension of the original data prior to any modeling is of the outmost interest. This reduction process directly modifies the dimension of any neural network or associative memory.

In mathematical terms, the problem under study can be stated as follows: given a p-dimensional random variable $\mathbf{x} = (x_1,\ldots,x_p)^T$, find a lower dimensional representation of it, $\mathbf{s} = (s_1,\ldots,s_k)^T$ with $k < p$, that captures the content in the original data, according to some criteria, usually stated within a supervision/learning/training set.

Several statistical techniques, such as principal component analysis [11], [13] and factor analysis have been proposed for achieving this dimensional reduction. Although these techniques based on second-order statistics are computationally expensive they are widely used. In other cases, when dealing with statistically normal variables (those with mean $= 0$) the covariance matrix already contains all the necessary information about the data. Second-order methods for dimensional reduction are relatively easy to code, as they require only simple matrix operations. However, many datasets of interest are not suitable for studying within a Gaussian distribution. For those cases, higher-order dimensional reduction methods, using information not contained in the covariance matrix, are more appropriate. Examples of these methods are independent component analysis and projection pursuit [20]. Another interesting method is random projections, which is a simple, yet powerful dimensional reduction technique that uses random matrices to project data into lower dimensional spaces [12], [15], [17], [21].

Some very useful non-statistical methods for dimensional reduction were proposed in the mid-fifties in the former Soviet Union and were later developed in other Eastern European countries and in Cuba. These logical-combinatorial methods are based on Testor Theory (formerly referred to as "Test Theory") and use the concepts of Testor and Non-reducible Testor [24] which were introduced for the first time by Yablonskii and Cheguis [1], [2] and later applied to classification problems by Dimitriev et al [5]. Testor Theory methods are used in this research to reduce the dimensionality of the data taken as input to different neural network models.

3 Dimensional Reduction Using Testor Theory

In this section, some aspects of Testor Theory applied to feature selection and dimensional reduction problems are applied. We also describe a technique, devised by Godoy-Calderón et al [25], used to compute a set of special kind of Testors called Super-Testors used to reduce the dimension of a neural network model.

3.1 Basics of Testor Theory

This section is based on [30]. In the framework of the logical-combinatorial pattern recognition [16], [19], [22], feature selection or dimensional reduction could be made by using Non-reducible Testors [23]. If $\Re$ is the whole set of attributes of the objects under study and thus the corresponding patterns are $\Re$-dimensional, a Testor is defined as follows:

Definition 1. A feature subset $\tau \subseteq \Re$ is a Testor if and only if when all features, except those from τ, are eliminated from the descriptions, no pair of similar subdescriptions remain in different classes. This definition indicates that a Testor is a feature subset, which allows complete differentiation of objects from different classes. Within the set of all Testors, there are some which are Non-reducible. These kind of Testors are called Typical Testors and are defined as follows:

Definition 2. A feature subset $\tau \subseteq \Re$ is a Typical Testor if and only if τ is a Testor and there is no other Testor τ' such that $\tau' \subset \tau$. This definition indicates that a Typical Testor is a Testor where every feature is essential, this is, if any of them are eliminated the resultant set is not a Testor.

The dimensional reduction approach based on Testor Theory was first proposed by Dimitriev [5] and the basic idea is the following: A Testor is a feature subset, which does not induce confusion between any pair of sub-descriptions of objects from different classes. Moving, from a Testor to a Typical Testor (eliminating features, when it is possible) we get an irreducible combination of features, where each feature is essential in order to keep differences between classes.

3.2 Testors and fuzzy classification of objects.

Let O be a set of objects denoted by o_i, each object is described in terms of a set of n features denoted by $\mathbf{x}^i = \delta(o_i)$ and these objects are grouped into c classes. Let $M = [m_{ik}]_{p \times c}$ be the membership matrix where p is the number of descriptions, c the number of classes and $m_{ik} = \mu_k(\delta(o_i))$ the membership of object o_i to class k given by:

$$\mu_k(o_i) = \bigvee_{j=1}^{p} \left(\beta\left(\delta(o_i), \delta(o_j)\right) * \mu_k\left(\delta(o_j)\right) \right) \tag{1}$$

where $\beta\left(\delta(o_i), \delta(o_j)\right)$ is any similarity function between o_i and o_j which yields a result in the range [0,1]. Confusion between objects in different classes depends on their membership to each class. An object o_i is confused between two classes a and b if and only if $\delta(o_i) \in (a \cap b)$.

Definition 3. The Discrimination Error ε of an object o_i is given by:

$$\varepsilon\left(\delta(o_i)\right) = \sum_{i,j \in [1,c]} \mu_{a_i \cap b_j}\left(\delta(o_i)\right) \tag{2}$$

this is the sum of its membership to any intersection between classes. The Cumulative Discrimination Error $\hat{\varepsilon}$ is given by:

$$\hat{\varepsilon} = \sum_{i=0}^{p} \varepsilon\left(\delta(o_i)\right) \tag{3}$$

Definition 4. A feature subset $\tau \subseteq \Re$ is a Testor with Level n if and only if $\hat{\varepsilon} = \sum_{i=0}^{p} \varepsilon\left(\delta(o_i)\big|_{\tau}\right) = n$. When $n = 0$ this definition is equivalent to Definition 1.

Definition 4 allows any subset of $\Re$ to be a Testor but with a different level. Of course, the most interesting Testors are those which have level zero.

4 Dimensional Reduction of a Neural Network Model

Now we will show how a neural model, which solves a supervised classification problem, can be optimized by using Super-Testors. In this paper several neural network models used for solving object recognition problems. as in [31], were optimized.

4.1 Associative Memories

Let $\left(x^{\xi}, i\right)_{\xi=1}^{p}, x^{\xi} \in \Re^n, i = 1, \ldots, c$ be a set of p-fundamental couples (SFC) formed by a pattern x^{ξ} and its corresponding class-index i. We want to build an associative memory **M**, using this SFC, that allows us to classify the patterns into their corresponding classes, i.e. $\mathbf{M} \otimes x^{\xi} = i$ for ξ=1,..., p and which, even in the presence of

distortions, classifies them adequately, i.e. $\mathrm{M} \otimes \tilde{x}^{\xi} = i$, where $\tilde{x}^{\xi}$ is an altered version of x^{ξ}. Operator $\otimes$ is chosen such that, when pattern x^{ξ} is operated with matrix **M**, it produces the corresponding index class of pattern x^{ξ}.

The associative memory **M** is built in terms of a ϕ function as follows:

$$\mathrm{M} = \begin{bmatrix} \phi_1 \\ \vdots \\ \phi_c \end{bmatrix} \tag{4}$$

where each ϕ_i represents the *i*-th row of a matrix **M** and this function is a codification of all patterns belonging to class *i*. In this case ϕ_i is defined as:

$$\phi_i^j = \frac{\gamma_i^j + \lambda_i^j}{2} \tag{5}$$

where

$$\gamma_i^j = \bigvee_{\xi=1}^{p} \left(x_i^{\xi,j} \right) \tag{6}$$

and

$$\lambda_i^j = \bigwedge_{\xi=1}^{p} \left(x_i^{\xi,j} \right) \tag{7}$$

i stands for the object's class and j goes from 0 to n. the size of the pattern. It can be seen that the idea is to build a hyper-box enclosing patterns that belong to class *i*, by means of max "$\vee$" and min "$\wedge$" set operators.

Once the associative memory is trained, pattern classification is done as follows: Given a new pattern $x^{\xi} \in \Re^n$ the index class i is given as

$$i = \mathrm{M} \otimes x^{\xi} = \arg \min_{l} \left[\bigwedge_{l=1}^{m} \bigvee_{j=1}^{n} \left| m_{lj} - x_j \right| \right] \tag{8}$$

Operators $\vee \equiv \max$ and $\wedge \equiv \min$ execute morphological operations on the difference of the absolute values of element m_{lj} of M and the components x_j of pattern x^{ξ} to be classified. $\bigvee_{j=1}^{n} \left| m_{lj} - x_j \right|$ is a metric formed with the maximum between row l of M and pattern x^{ξ}, thus it can be written as $d(x, m_l) \equiv \bigvee_{j=1}^{n} \left| m_{lj} - x_j \right|$, m_l row of

$\mathbf{M}$. With this metric, pattern classification is the process of assigning pattern x^{ξ} to the class whose row index is the nearest.

4.2 Optimizing the Associative Memory

The SFC can be seen as the set of objects denoted by o_i with cardinality p; each object is described in terms of a set of n features denoted by $\mathbf{x}^i = \delta(o_i)$ and grouped into c classes. Optimization of the associative model is done by computing the cumulative error of each testor τ, as shown in the next algorithm:

1. Select a feature subset τ.
2. Compute membership matrix as in section 3.2, using equation 1 with Testor τ.
3. Compute accumulative error using equation 3.
4. Go to step 1 until the cumulative error of the whole feature subset τ has been computed.
5. Select feature subset τ, where level n of the Testor is the minimum.
6. Finally with this feature subset τ, train the associative memory.

5 Experimental Results

In this section, the proposal is tested with the set of realistic objects shown in Figure 1. Objects were not directly recognized by their images but instead from their invariant descriptions. The associative memory $\mathbf{M}$ is built with these invariant descriptions. Twenty images of each object in different positions, translations and scaled changes were used to get the invariant descriptions.

(a) (b) (c) (d) (e)

Fig. 1. The five objects used for training the associative memory. (a) A bolt, (b) A washer, (c) An eyebolt, (d) A hook, (e) A dovetail

A standard threshold [8] was applied to each of the 20 images in order to get their binary version. Small spurious regions from each image were eliminated by means of a standard-size filter [14]. Next, for each of the 20 images of each object (class) seven well-known Hu geometric moments invariant to translations, rotations and scale changes, were computed [4]. After applying the methodology described in Section 4.1, the associative memory $\mathbf{M}$ is:

$$M = \begin{bmatrix} 0.4394 & 0.1598 & 0.0071 & 0.0028 & 1.96E-5 & 0.0011 & -8.47E-6 \\ 0.1900 & 8.72E-5 & 7.47E-6 & 1.28E-14 & 7.23E-14 & -2.93E-10 & -1.6E-14 \\ 0.7092 & 0.2895 & 0.1847 & 0.0730 & 0.0088 & 0.0394 & -0.0015 \\ 1.4309 & 1.6009 & 0.7944 & 0.2097 & 0.0831 & 0.1565 & 0.0118 \\ 0.2475 & 0.0190 & 2.5E-5 & 8.66E-5 & 4.82E-9 & 1.20E-5 & -1.4E-9 \end{bmatrix}$$

A new set of images was used to test the efficiency of the proposal. This set consisted of 100 images shown in Figure 2 (20 for each of the five objects), different from those used to get the associative memory M. Using this memory all objects of the set of images were put in their corresponding class. Thus, the performance of the proposal was 100%.

(a) (b) (c) (d) (e)

Fig. 2. The five objects used in the experiments: (a) A bolt. (b) A washer. (c) An eyebolt. (d) A hook. (e) A dovetail

In order to optimize the associative memory and reduce its dimensionality we applied the algorithm described in section 4.2. By applying this algorithm, we found that the feature subset τ, where level n of the testor is the minimum, was $\tau_{64} = \{x_1\}$, $\tau_{32} = \{x_2\}$ and $\tau_{96} = \{x_1, x_2\}$. After applying methodology described in Section 4.1, the corresponding associative memories are defined as:

$$M_{64} = \begin{bmatrix} 0.4394 \\ 0.1900 \\ 0.7092 \\ 1.4309 \\ 0.2475 \end{bmatrix} \quad M_{32} = \begin{bmatrix} 0.1598 \\ 8.72E-5 \\ 0.2896 \\ 1.6009 \\ 0.0190 \end{bmatrix} \quad M_{94} = \begin{bmatrix} 0.4394 & 0.1598 \\ 0.1900 & 8.72E-5 \\ 0.7092 & 0.2896 \\ 1.4309 & 1.6009 \\ 0.2475 & 0.0190 \end{bmatrix}$$

Using associative memories M_{64} and M_{94} all objects of the set of images were put in their corresponding class. Thus, performance of the two optimized associative memories was of 100%. For the case of the associative memory M_{32} the performance was reduced to 84 %,.This result indicated that second moment of Hu is less representative than first moment . Table 1 summarizes the classification results for all associative memories tested.

Table 1. Comparative classification percentages of the associative memory with respect to the optimized associative memories

	M	M_{32}	M_{64}	M_{96}
Bolt	100%	90%	100%	100%
Washer	100%	100%	100%	100%
Eyebolt	100%	65%	100%	100%
Hook	100%	100%	100%	100%
Dovetail	100%	85%	100%	100%

In order to demonstrate that this technique is independent of the network architecture used, we performed several experiments using other neural networks. The same experiments were performed using the associative memories described in [32] and [27]. These associative memories use three different operators: **prom** operator, **med** operator and **median** operator. On the other hand we used the well-known multilayer neural network trained with the back-propagation algorithm as described in [10].

By applying the algorithm described in section 4.2, we found that the feature subset τ, where level n of the Testor is the least, was $\tau_{64} = \{x_1\}$, $\tau_{32} = \{x_2\}$ and $\tau_{96} = \{x_1, x_2\}$ for the different neural models.

For the case of **prom** operator, by using associative memories M_{64} and M_{94} all objects of the set of images were put in their corresponding class. Thus, the performance of the two optimized associative memories was of 100%. For the case of the associative memory M_{32} the performance was reduced to 88 %. Table 2 summarizes the classification results for all associative memories tested trained with **prom** operator.

Table 2. Comparative classification percentages of the associative memory using **prom** operator with respect to the optimized associative memories

	M	M_{32}	M_{64}	M_{96}
Bolt	100%	90%	100%	100%
Washer	100%	100%	100%	100%
Eyebolt	100%	65%	100%	100%
Hook	100%	100%	100%	100%
Dovetail	100%	85%	100%	100%

For the **med** operator case, by using associative memories M_{32}, M_{64} and M_{94} the performance was 82 %, which is the same result obtained with the complete description (e.g. with M). Table 3 summarizes the classification results for all associative memories tested and trained with **med** operator.

Table 3. Comparative classification percentages of the associative memory using **med** operator with respect to the optimized associative memories

	M	M_{32}	M_{64}	M_{96}
Bolt	100%	100%	100%	100%
Washer	100%	100%	100%	100%
Eyebolt	70%	70%	70%	70%
Hook	50%	50%	50%	50%
Dovetail	90%	90%	90%	90%

For the case of the **median** operator, by using associative memories M_{32}, M_{64} and M_{94} the performance was 95 %, again the same result obtained without reducing the network. Table 4 summarizes the classification results for all associative memories tested trained with **median** operator.

Table 4. Comparative classification percentages of the associative memory using **median** operator with respect to the optimized associative memories

	M	M_{32}	M_{64}	M_{96}
Bolt	100%	100%	100%	100%
Washer	100%	100%	100%	100%
Eyebolt	90%	90%	90%	90%
Hook	50%	50%	50%	50%
Dovetail	85%	85%	85%	85%

For the case of the multilayer neural network trained with the back-propagation algorithm, the performance for M_{32}, M_{64} and M_{94} was 89 %, 100% and 100%, while for M it was 98%. Table 5 summarizes the classification results for all neural networks trained with the back-propagation algorithm.

Table 5. Comparative classification percentages of the neural network with respect to the optimized neural network.

	M (7-4-1)	M_{32} (1-4-1)	M_{64} (1-4-1)	M_{96} (2-4-1)
Bolt	100%	95%	100%	100%
Washer	95%	100%	100%	100%
Eyebolt	95%	70%	100%	100%
Hook	100%	95%	100%	100%
Dovetail	100%	85%	100%	100%

6 Conclusions

In this paper we have described a new technique to optimize the size of an associative memory in a object recognition problem. This technique uses Testor theory, widely used in the logical-combinatorial approach to pattern recognition.

We describe an algorithm which allows us to calculate the most relevant features in the patterns used to train neural network models as associative memories and multilayer neural networks.

Throughout several experiments we test the accuracy of the proposal by using images of real objects. In those experiments first we train the neural models and then we show that by applying the procedure described in section 4.2 we can reduce the size of the neural models. We also show that in some cases the accuracy of the models is increased; like in the multilayer neural network, and in other cases the accuracy is the same. We are presently working in a comparison between classical dimensional reduction techniques and Testor Theory techniques applied to the neural network optimization task and also testing this approach with more complex objects.

Acknowledgments. Authors are grateful for the economic support provided by SIP-IPN under grant 20071432.

References

1. I.A. Cheguis, S.V. Yablonskii (1955), About testors for electrical outlines. *Usp. Mat. Nauk*, 4(66): 182-184 (in Russian).
2. I.A. Cheguis, S.V. Yablonskii(1958), Logical methods for controlling electrical systems. *Trudy MIAN ime V.A. Steklova*, LI :270-360 (in Russian).
3. K. Steinbuch (1961). Die Lernmatrix. *Kybernetik*, 1(1):26-45.
4. M. K. Hu (1962). Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8:179-187 .
5. Dimitriev A. N., Zhuravlev Y.I. and Krendeliev F.P. (1966), "About mathematical principles of objects and phenomena classification", *Diskretni Analiz* 7 , (In Russian).
6. J. A. Anderson (1972). A simple neural network generating an interactive memory. *Mathematical Biosciences*, 14:197-220.
7. T. Kohonen (1972). Correlation matrix memories. *IEEE Trans. on Computers*, 21(4):353-359.
8. N. Otsu (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on SMC*, 9(1):62-66.
9. J. J. Hopfield (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79: 2554-2558, 1982.
10. D. Rumelhart and J. McClelland (1986). Parallel distributed processing group. MIT Press.
11. I..T. Jolliffe(1986). *Principal Component Analysis*. Springer-Verlag.
12. H. Ritter and T. Kohonen (1989). Self-organizing semantic maps. *Biologial Cybernetic*, 61:241-251.
13. J.E. Jackson (1991). *A User's Guide to Principal Components*. New York: John Wiley and Sons.
14. R. Jain et al. *Machine Vision* (1995). McGraw-Hill. pp. 47-48.
15. S. Kaski (1997). *Data exploration using self-organizing maps*. PhD thesis, Helsinki University of Technology, Finland.

16. Alba Cabrera E. (1997), "*New extensions of testor concept for different types of similarity functions*" Ph. D. Thesis, ICIMAF, Cuba. (In Spanish).
17. S. Kaski (1998). Dimensionality reduction by random mapping: fast similarity computation for clustering. *Proc. IEEE International Joint Conference on Neural Networks*, 1:413-418.
18. R. Kohavi and G. John (1998). The wrapper approach. In H. Liu and H. Motoda, editors, *Feature Extraction, Construction and Selection: A Data Mining Perspective.* Springer Verlag.
19. J. Ruiz Shulcloper, A. Guzmán Arenas, and J. F. Martínez Trinidad (1999), *Logical Combinatorial Approach to Pattern Recognition. Feature Selection and Supervised Classification*, IPN, Mexico(In Spanish).
20. A. Hyvarinen (1999). Survey on independent component analysis. *Neural Computing Surveys*, 2:94-128.
21. T. Kohonen et al (2000). Self organization of massive document collection. *IEEE Transactions on Neural Networks*, 11(3):574-585.
22. J. F. Martínez Trinidad, and A. Guzmán-Arenas (2001), "The logical combinatorial approach to pattern recognition an overview through selected works", *Pattern Recognition*, 34(4):741-751.
23. M. Lazo-Cortes, J. Ruiz-Shulcloper and E. Alba-Cabrera (2001), "An overview of the evolution of the concept of testor", *Pattern Recognition*, 34(4):753-762.
24. J. Ruiz Shulcloper, and M.A. Abidi (2002). Logical Combinatorial Pattern Recognition: A Review. Recent Research Developments in Pattern Recognition, Pub. Transword Research Networks, USA, 133-176
25. S. Godoy-Calderón, M. Lazo-Cortés, J.F. Martínez-Trinidad (2003). A non-classical view of Coverings and its implications in the formalization of Pattern Recognition problems. WSEAS Transactions on Mathematics, 2, 1-2, 60-66.
26. P. Sussner (2003). Generalizing operations of binary auto-associative morphological memories using fuzzy set theory. *Journal of mathematical Imaging and Vision*, 19(2):81-93.
27. G. X. Ritter, G. Urcid, L. Iancu (2003). Reconstruction of patterns from noisy inputs using morphological associative memories. *Journal of mathematical Imaging and Vision*, 19(2):95-111.
28. H. Sossa and R. Barron (2003). New associative model for pattern recall in the presence of mixed noise. *In Proceedings of the fifth IASTED International Conference on Signal and Image Processing*, SIP2003. Acta Press 399:485-490.
29. H. Sossa, R. Barrón, R. A. Vázquez (2004). Transforming Fundamental set of Patterns to a Canonical Form to Improve Pattern Recall. *In proceedings of Ninth Ibero-American Conference on Artificial Intelligence* (IBERAMIA2004), LNAI 3315:687-696.
30. José Á. Santos, Ariel Carrasco and José F. Martínez (2004). Feature Selection using Typical Testors applied to Estimation of Stellar Parameters. *Computación y Sistemas*, 8(1):15-23.
31. R. A. Vázquez, H. Sossa and R. Barrón (2005). Object classification based on associative memories and midpoint operator. *Advances in Artificial Intelligence Theory* RCS 16:131-140, CIC-IPN.
32. R. A. Vázquez, H. Sossa (2006). Image categorization using associative memories. *Advances in Artificial Intelligence*, LNAI: 367-380.
33. R. A. Vázquez, H. Sossa and B. A. Garro (2006). A new bi-directional associative memory. *Progress in Pattern recognition, image analysis and application*, LNCS:549-558.
34. H. Sossa, R. Barrón and R. A. Vázquez (2004). *Real valued pattern classification based on extended associative memories.* Proceedings of 5th Mexican International Conference on Computer Science (ENC 2004), 20-24. IEEE Computer Society, edited by R. Baez, J. Marroquin and E. Chavez, pp 213-219.

Feature Extraction using Gabor and DWT for GMM based Face Verification Algorithms

Jesus Olivares-Mercado, Gabriel Sanchez-Perez, Mariko Nakano-Miyatake, Hector Perez-Meana

Postgraduate Section of Mechanical Electrical Engineering School, Instituto Politécnco Nacional, Av. Santa Ana no. 1000 Col. San Francisco Culhuacan, Mexico D.F. Mexico
hmpm@prodigy.net.mx

(Paper received on July 01, 2007, accepted on September 1, 2007)

Abstract. This paper presents two feature extraction methods for developing face verification algorithms based on the Gaussian Mixtures Model (GMM). The first one using the discrete wavelet transform, (DWT) while the second is based on the discrete Gabor transform (DGT). In both cases, firstly the feature extraction is carried out using either the DGF or the DWT. Next the Gaussian Mixture Model (GMM) is used to perform the face verification task. Evaluation results using the standard data bases with different parameters, such as the mixtures number, the number of faces used for training as well as the transform used for feature extraction show that proposed system provides better results that other previously proposed systems with a correctly detections larger than 95%, using any of these transforms,. Although, as happens in must face recognition systems, the verification rate decreases when the target faces present some rotation degrees.

1 Introduction

The development of security systems based on biometric features is currently a topic of active research, because it has a great importance in the development of the identity verification systems for access control, to enforce the security in restricted areas, and several other security applications. The terrorist attacks that have happened during the last decade have done evident the necessity of developing more reliable security systems, in offices, banks, companies, trades, etc. Among them the identity verification based on biometric methods appear to be a good alternative for the development of such security systems.

The biometrics systems consist of a group of automated methods for recognition or verification of people identity using physical characteristics or personal behavior of the person under analysis [1]. This technology is based on the fact that each person is unique and possesses distinctive features that can be used to identify her/him. Following these ideas several biometric based security systems have been developed using fingerprints, iris, voice hand and face features. Among them, the face verification systems appear to be a desirable alternative because in is non-invasive and its computational complexity is low, it is the biometric method easier of understanding

© H. Sossa, R. Barrón, E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 105-114

since for us the face is the most direct way to identify people; besides that the data acquisition of this method consists on taking a picture, doing it one of the biometric methods with larger acceptance among the users.

The recognition is a very complex activity of the human brain. For example, we can recognize hundred of faces learned throughout our life and to identify familiar faces at the first sight even after several years of separation with relative easy. However for a computer it is not a simple task. For instance, recently proposed face recognition systems, achieve a recognition rate of about 91% when the face image is not rotated or the rotation is relatively low. However although, this recognition rate is good enough for several practical applications, it may be so large for applications where the security should be extreme; such that we cannot tolerate a high erroneous recognition average. This paper proposes a face recognition algorithm that is able of achieving an erroneous verification rate below 9%. Several methods have been proposed for face recognition [2], [3] such as the methods based on statistical correlation of the geometry [4]; the face form which uses the distances among the position of the eyes, mouth, nose, etc. as well as those using the neuronal networks technology that trait to imitate the operation of the human brain [2]. Many of these systems can recognize a person even when they present some physical changes, such as the growth of the beard or mustache, changes in the color or the style of the hair, the use of glasses, etc. Although in general are sensitive to rotations of the face images.

Before starting the proposed methods analysis used for face recognition, it is necessary to point out the verification concept. In face verification, the person says to the system about his/her identity, presenting an identification card or writing a special password. The system holds the person's features (for example the persons face in this case), and then proceeds to solve if the person is who (his/her) claims to be.

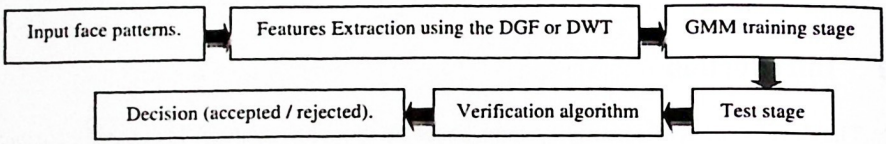


Fig. 1. Proposed face verification algorithm.

2 Proposed System

This section provides a detailed description of the proposed face verification algorithm which consists of three stages. Firstly a feature extraction of the face is carried out, using either the Gabor discrete transform (DGF) or discrete wavelet transform (DWT). Next using these features vectors, a model for each face is obtaining using a Gaussian Mixtures Model (GMM). Finally during the verification process, the GMM output is used to take the final decision. Figure 1 shows the block diagram of proposed algorithm.

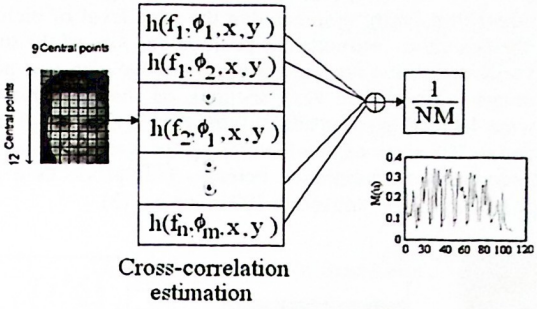


Fig. 2. DGT model.

2.1 Feature extraction with DGT

The feature extraction stage is one of the most important steps in any pattern recognition system. To this end, the proposed algorithm uses the DGT which has some relation with the human visual system (HVS). The two dimensional discrete Gabor functions (2D-DGF) depends on four parameters, two of them express their localization in the space (x, y) while the other two express the spatial frequency, f_m, and the orientation ϕ_n, where $m=1,2,..N_f$ and $n=1,2,...,N_\phi$[5]. Thus to estimate the features vector, firstly the captured image (NxM) is divided in M_xM_y receptive fields each one of size $(2N_x+1)x(2N_y+1)$ (Fig. 2), where $N_x=(N-M_x)/2M_x$, $N_y=(M-M_y)/2M_y$. This fact allows that the number of elements in the features vector be independent of the captured image size. Next, the central point of each receptive field whose coordinates are given by (c_i,d_k), where $i=1,2,..,N_x$; $k=1,2,3,...,N_y$ are estimated. Subsequently the first point of the cross-correlation $\psi(u,v)$ between each receptive field and the N_fN_ϕ Gabor functions $h_{m,\phi}(x,y)$ is estimated, where

$$h_{f,\phi}(x,y) = g(x',y')\exp(2\pi j f_m(x'+y')) \tag{1}$$

where denotes the Gabor function, and

$$(x',y') = ((x\cos\phi_n + y\sin\phi_n),(-x\sin\phi_n + y\cos\phi_n)) \tag{2}$$

As shown in Fig. 1, N_fN_ϕ correlations are estimated for each receptive field, leading to an extremely large features vector. Thus to reduce the elements in the features vector, the first point of the total cross correlation between each receptive field and the set of DGF is used, which can be obtained taking the average of $\psi(u,v)$ with respect to v. Therefore the proposed algorithm features vector $M(u)$, is given by

$$M(u) = \frac{1}{N_v}\sum_{v=1}^{N_v}|\psi(u,v)| \tag{3}$$

where $N_v=N_fN_\phi$. To do the proposed method robust against changes of sizes and translation; the algorithm firstly assumes that the gray level of picture background is constant. Next the algorithm estimates the position and size of the image by analyzing the gray label variation on the image. Once the image size and position have been estimated, the image is divided in 12x9 sections, as shown in Fig. 2, whose central point will be always located in the space position (x, y), where x=0 and y=0. After the image was divided in 108 sections, the features vector was estimated with 9 phases and 6 normalized frequencies as mentioned before. This produces a matrix with 5832 elements that are subsequently reduced to 108 using eq. (3).

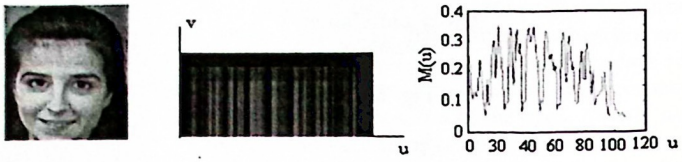


Fig. 3. a) Original image. b) Cross correlation matrix between DGF and receptive fields. c) Features vector obtained of proposed algorithm.

2.2 Feature extraction with DWT

The most commonly used set of discrete wavelet transforms was formulated by the Belgian mathematician Ingrid Daubechies in 1988. This formulation is based on the use of recurrence relations to generate progressively finer discrete samplings of an implicit mother wavelet function; each resolution is a half of that of the previous scale.

The DWT of a given signal x is estimated by passing it through a series of low pass and high pass filters (Fig. 4. First the samples are passed through a low pass filter with impulse response $g(n,m)$ resulting in a convolution of the two. The signal is also decomposed simultaneously using a high-pass filter $h(n,m)$. The detail coefficients are the high-pass filter outputs and the approximation coefficients are the low-pass ones. It is important that the two filters, related to each other, are known as a quadrature mirror filter. However, since half the frequencies of the signal have now been removed, half the samples can be discarded according to Nyquist's rule. The filter outputs are:

$$Y_{LOW}(n,m) = \sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} x(n,m)g(2n-k,2m-j) \tag{4}$$

$$Y_{HIGH}(n,m) = \sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} x(n,m)g(2n-k,2m-j) \tag{5}$$

This decomposition reduces the spatial resolution since only a quarter of each filter output allows characterizing the face image. However, because each output has band width equal to a quarter of the original one, the output image can be decimated to reduce the image size.

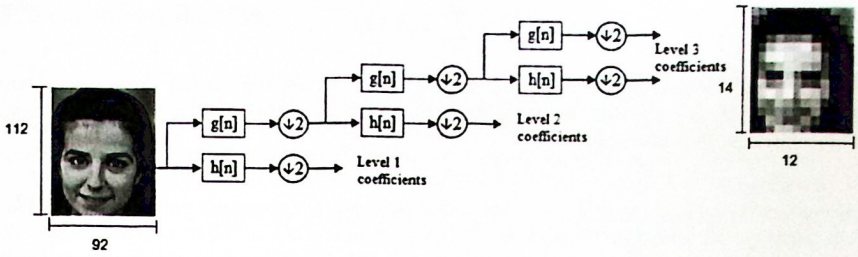


Fig 4. 3 level wavelet decomposition

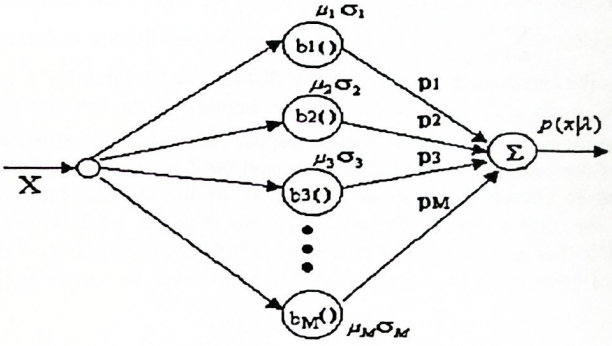


Fig. 5. Gaussian Mixture Model

Here only the approximation coefficients are used to characterize the face image. This decomposition is repeated to further increase the frequency resolution and the approximation coefficients decomposed with high and low pass filters and then down-sampled. This is represented as a binary tree with nodes representing a sub-space with different time-frequency localization. The tree is known as a filter bank.

At each level in the above diagram the signal is decomposed into low and high frequencies. Due to the decomposition process the input signal must be a multiple of 2^n where n is the number of levels. For example a signal with 32 samples, frequency range 0 to f_n and 3 levels of decomposition, 4 output scales are produced:

2.3 Face verification stage

To perform the face verification task a GMM will be used because, the GMM, which consists of a sum of M weighted Gaussian density functions, is able to approximate any probability distribution if the number of Gaussian components is large enough. Consider the GMM shown in Fig. 5 which is described by the following equation [6]:

$$p(\vec{x} / \lambda) = \sum_{i=1}^{M} p_i b_i(\vec{x}) \tag{6}$$

where $\vec{x}$ is a N-dimensional vector, $b_i(\vec{x}), i=1,2,...M$, are the density components and p_i, $i=1,2,...M$, are the mixture weights. Each density component is a D-dimensional Gaussian function given as:

$$b_i(\vec{x}) = \frac{1}{(2\pi)^{D/2}|\sigma_i|^{1/2}} \exp\left\{-\frac{1}{2}(\vec{x} - \vec{\mu}_i)' \sigma_i^{-1} (\vec{x} - \vec{\mu}_i)\right\} \tag{7}$$

Where ( )' denotes the transpose vector, μ_i denotes the mean vector and σ_i the covariance matrix which is assumed to be diagonal; and p_i are the mixture weights which satisfies that $\sum_{i=1}^{M} p_i = 1$. The model of the distribution is determined by the mean vector, the covariance matrix and the distribution weights for a face model is given by $\lambda = \{p_i, \mu_i, \sigma_i\}$ $i = 1,2,..,M$. The estimation of the optimal parameter is a non-linear problem, such that we need an iterative algorithm to estimate the optimal parameters of the face verification algorithm, and the Maximum Likelihood algorithm (ML) is used to search the optimal parameters of the system, providing the best approach to the face model under analysis. So then the goal is to find the best parameters of λ that maximize the a posteriori probability distribution. For a sequence of vectors T of training $X=\{x_1,...,x_T\}$, GMM likelihood can be written as [7]:

$$p(X / \lambda) = \prod_{t=1}^{T} p(X_t / \lambda) \tag{8}$$

Equation (8) is not lineal in relation of the parameters λ, then is necessary to carry out the estimation in an iterative way using the Expectation-Maximization algorithm (EM), in which starting from an initial set of parameters $\lambda(r-1)$ and a new model is estimated $\lambda(r)$, where r denotes the r-th iteration, so that:

$$p(X / \lambda(r)) \geq p(X / \lambda(r-1)) \tag{9}$$

To achieve this goal, each T partial feature vectors $\mathbf{X}_t$, of the GMM parameters are updated [7]. During the testing phase we need to estimate the probability that one face under analysis corresponds to a given model, that is $P_r(\lambda/X)$. To achieve this, the Bayes theorem is used, then obtain:

$$\hat{R} = \sum_{t=1}^{T} \log_{10}(p(X_t / \lambda)) \tag{10}$$

where $p(X/\lambda)$ is the conditional probability of the face $\mathbf{X}$ given by face model λ, this is the GMM system response shown in the Fig. 6.

3 Evaluation Results

The evaluation of proposed system was carried out by computer simulations using the database created by Olivetti Research Laboratory in Cambridge, UK (ORL), which consists of images of 30 people with 10 images of each one which differs on, face rotation, different inclination, etc. The images have a size of 92 x 112 pixels.

After the feature vector is estimated using the DGT or the DWT it is applied to a GMM for obtaining the model of the face under analysis. This is achieved introducing the estimated vectors to the Gaussian mixture model (GMM) to obtain the weights, the mean and variance as described in [6], [7], where for training we assume that the 108 elements obtained in the feature extraction stage are divided in partial features vector of L elements as follows

$$S = \{X_0, X_1, X_2, ..., X_T, X_{T+1}, X_{T+2}, X_{T+3}, ..., X_{L-1}\} \tag{11}$$

Subsequently, form a group of vectors in L segments with T features vectors, Xt, each one in the following way:

$$S_0 = \{X_0, X_1, X_2, X_t, X_{t+1}, ..., X_T\} \tag{12}$$

$$S_k = \{X_k, X_{k+1}, X_{k+2}, ..., X_t, X_{t+1}, ..., X_{T+k}\} \tag{13}$$

In this work the system uses T=12 in order that each 12 feature vectors of the GMM parameters being updated. In the table 1 and 3 the face rejection results using the DGT DWT respectively are shown, the Fig. 2 and 4 show the false acceptation rate using DGT and DWT respectively.

Table 1. False rejection error using DGT.

# of faces of training	# Gaussian Mixtures	Low threshold %	Half threshold %	high threshold %
1 face	9	12	8	5.1
	16	18.5	12.9	8.7
	32	12.4	9.2	6.8
3 faces	9	12	7.7	4.7
	16	13.25	9.1	5.9
	32	8.66	6.48	4.6
5 faces	9	9.3	5.9	4
	16	10.9	7	4.4
	32	5.9	4.6	3.9

Table 2. False acceptation error using DGT.

# of faces of training	# Gaussian Mixtures	Low threshold %	Half threshold %	high threshold %
1 face	9	11.57	6.71	3.14
	16	18.5	12.35	7.7
	32	11.8	8.34	5.62
3 face	9	11.4	6.5	3.13
	16	12.5	8	4.5
	32	7.42	5	2.9
5 face	9	8.7	4.5	2.2
	16	10.2	5.8	2.9
	32	4.4	2.8	2

Table 3. False rejection error using DWT.

# of faces of training	# Gaussian Mixtures	Low threshold %	Half threshold %	high threshold %
1 face	12	4.55	4	3.41
	16	5.53	4.2	3.3
	32	9.8	7.9	6.1
3 faces	12	1.5	2.18	2.26
	16	8.8	7.32	6.08
	32	16.7	14.8	13.3
5 faces	12	6.7	5.9	5.3
	16	3.6	3.2	2.9
	32	3.4	3	2.9

Table 4. False acceptation error using DWT.

# of faces of training	# Gaussian Mixtures	Low threshold %	Half threshold %	high threshold %
1 face	12	3.82	2.63	1.74
	16	4.54	3	1.9
	32	9.2	7.11	5.1
3 face	12	0.66	0.56	0.43
	16	8	6.45	5.11
	32	16.3	14.3	12.7
5 face	12	5.8	4.3	3.34
	16	1.31	0.9	0.44
	32	1.45	1	0.8

In the verification stage a threshold is used which depends of the face under analysis. Top improve the verification performance this threshold may be divided in three categories: low, half and high thresholds. Two variants more were introduced for evaluation: one is the numbers of faces used for the training and the second one is the numbers of Gaussians mixtures to be used and these variants are applied in the two used techniques

Simulation results show that proposed algorithm performs fairly well in comparison with other previously proposed methods [1], [2], [6], even with faces that present an appreciable rotation, as happens in the ORL database. We can also see that there is not a much difference between using DWT and DGT.

4 Conclusions

This paper presented two face verification algorithms in which the DGT or DWT are used for feature extraction and the GMM to perform the verification task. Evaluation results obtained were very different with each one of the variants proposals, from 18.5% in the worst case, until 3.9% in the best case, using DGT, and 16.7% in the worst case, until 2.9% in the best case, using DWT. These results are very satisfactory if we consider that the database used is composed by 30 persons. This quantity of face is very similar to any database in a real application. In the case of accepting a person with a false identity we have a percentage of error of 18.5% at worst case and a 2% in the best one using the DGT; while using the DWT we have a 16.3% cases at worst case; and a 0.44% in the best one using DWT. In summary, can observe that the system performance becomes better when more faces are used for training the GMM and it has a larger number of mixtures. This is valid for false acceptation error as well as for false rejection error.

Acknowledgements

We thanks the National Science and Technology Council and to the National Polytechnic Institute for the financial support during the realization of this research.

References

1. P. Reid: BIOMETRICS for Networks Security, New Jersey: Prentice Hall, 2004, pp 3-7.
2. R. Chellappa, C. Wilson, and S. Sirohey: Human and Machine Recognition of Faces: A Survey, Proc. IEEE, vol. 83, no. 5, pp. 705-740, 1995.
3. A. Shashua: Geometry and Photometry in 3D Visual Recognition, PhD thesis, Massachusetts Institute of Technology, 1992.
4. Robert J. Baron: Mechanisms of human facial recognition. International Journal of Man-Machine Studies, pp: 137-178, 1981.
5. Dunn, D., Higgins, W. E.: Optimal Gabor Filters for Texture Segmentation, IEEE Trans. Image Proc., Vol. 4, No. 7, Jul. 1995.
6. Jin Y. Kim, Dae Y. Ko, Seung Y. Na.: Implementation and Enhancement of GMM Face Recognition Systems Using Flatness Measure, IEEE Robot and Human Interactive Communication, September 2004.
7. Douglas A. Reynolds, Richard C. Rose: Robust Text-Independent Speaker Identification Using Gaussian Mixture Speaker Models, IEEE Trans. Speech and audio Proc., Vol. 3, No. 1. Jan. 1995.

8. Busso C., Narayanan S. S.: Interrelation Between Speech and Facial Gestures in Emotional Utterances: A Single Subject Study, IEEE Transactions on Audio, Speech, and Language Processing, Vol. PP 2007.
9. Yan, S., Xu, D., Tang, X.: Face Verification With Balanced Thresholds, Transactions on Image Processing, IEEE, Vol. 16, No. 1, Pag. 262- 268, Jan 2007.
10. Kar, Soumitra, Hiremath, Swati, Joshi, Dilip G.: Chadda, Vinod.K.; Bajpai, Apurva, "A Multi-Algorithmic Face Recognition System", International Conference on Advanced Computing and Communications, Pag. 321 – 326, December 2006.

Perceptron for Feature Selection

Manuel Mejía-Lavalle[1], and Enrique Sucar[2]

[1]Instituto de Investigaciones Eléctricas, Reforma 113, 62490 Cuernavaca, Morelos, México
mlavalle@iie.org.mx
[2]INAOE L.E.Erro 1, 72840 StMa. Tonantzintla, Puebla, México
esucar@inaoep.mx

(Paper received on June 21, 2007, accepted on September 1, 2007)

Abstract. An exploratory and effective research around the possible application of the Perceptron paradigm as a method for feature selection is carried out. The main idea is training a Perceptron and then using the learned inter-connection weights as metric of which database's attributes are the most discriminative for the class. We hypothesized that an inter-connection weight close to zero indicates that the associated attribute to this weight can be eliminated because it does not contribute with relevant information in the construction of the class separator hyper-plane of the Perceptron. The experiments that were realized, with 8 real and 11 synthetic databases, show that the proposed algorithm is a good trade-off among feature reduction, accuracy and processing time.

1 Introduction

Feature selection has become a relevant and challenging problem for the area of Knowledge Discovery in databases (KDD). An effective feature selection strategy [1] can significantly reduce the data mining processing time, improve the predicted accuracy, and help to understand the induced models, as they tend to be smaller and make more sense to the user.

Although there is many feature selection algorithms reported in the specialized literature, none of them are perfect: some of them are fast, but not effective in the feature selection task (e.g., *filter* methods), and others are effective, but very costly in computational time (e.g., *wrapper* methods).

Specifically, filter methods are more efficient; they use some form of *correlation* measure between individual attributes and the class [2][3]; however, because they measure the relevance of each isolated attribute, they cannot detect if redundant attributes exist, or if a combination of two or more attributes, apparently irrelevant when analyzed independently, are indeed relevant together [4]. On the other hand wrapper methods, although effective in eliminating irrelevant and redundant attributes and detecting inter-dependencies, are very slow because they apply the mining algorithm many times, changing the number of attributes each time of execution as they follow some search and stop criteria [5].

Here, we propose and experiment with a feature selection strategy, in which the trained Perceptron inter-connection weights are utilized like a metric, indicator or measure of attribute importance or relevance. The basic idea is to eliminate the

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 115-125

attributes whose associated weights are close to zero. Similar idea is used in more complex, time-consuming and memory demanding methods, like the Principal Component Analysis (PCA) technique and the Support Vector Machine (SVM) variants for Feature Selection (SVM-FS) [1][7].

To cover these topics, the article is organized as follows: Section 2 introduces our feature selection method; Section 3 details the experiments; Section 4 discuss and surveys related work; finally, conclusions and future work are given in Section 5.

2 Feature Selection Perceptron Strategy

The method to explore is the classic Rosenblatt's Perceptron, as strategy for relevant attribute selection. We propose to use a "soft" or "relaxed" Perceptron (similar to [6]), in the sense that it can accept some percentage of misclassified instances; in this case the training stopping criterion occurs when no accuracy improvement is obtained. To measure the improvement/ no-improvement we use the generalization accuracy (Acc) and the Balanced Error Rate (BER) [7]. The Perceptron variation that we used has: a) as many inputs-inter-connection weights as attributes the dataset contain, b) only one neuron with a step activation function, and c) only one output. To obtain the Perceptron output *S*, we use the equation:

$$S = U\{\Sigma_i W_i E_{ij}\} \tag{1}$$

where W_i are the i inter-connection weights; E_{ij} is the input vector (with i elements) that form an instance j; and U is a step function that outputs *1* (one) if $\Sigma_i W_i E_{ij} > \theta$ and *0* (zero) otherwise. θ is the Perceptron's threshold.

To train the Perceptron we apply the equations:

$$W_i(t+1) = W_i(t) + \{\alpha(T-S)E_{ij}\} \tag{2}$$

$$\theta(t+1) = \theta(t) + \{-\alpha(T-S)E_{ij}\} \tag{3}$$

where T is the desired output (or target) and α is the learning rate: this user parameter can take values between 0 and 1.

Although there exist more sophisticated procedures in the area of neural network pruning [8], we choose this idea because of its simplicity (and therefore, efficiency) and direct application to feature selection, because of the direct relation between each feature and its Perceptron inter-connection weight. To execute the overall feature selection process we apply the procedure shown in Fig. 1.

3 Experiments

With the Perceptron for Feature Selection (PFS) we expect:

a) To use less amount of memory, because a Perceptron only requires to store as many inter-connection weights as "n" attributes the database has, as opposed to PCA that builds an "n^2" matrix,
b) To drop the processing time because, as opposed to SVM-FS [7] that involves solving a quadratic optimization problem, the Perceptron converges fast to an approximate solution,
c) To avoid to carry out a combinatorial explosive search or exhaustive exploration; that is to say, without having to evaluate multiple attribute subset combinations, as the wrapper methods do (they evaluate multiple subsets applying the same algorithm that will be used in the data mining phase), or some filter methods, that employ different metric to evaluate diverse attribute subsets, and that use a variety of search strategies like *Branch & Bound, Sequential Greedy, Best-First, Forward Selection, Sequential Backward Elimination, Floating, Random Search*, among others,
d) Implicitly capture the inter-dependences among attributes, as opposed to filter-ranking methods, that evaluate only the importance of one attribute against the class, like *F-score, Symmetrical Uncertainty, Correlation, Entropy, Information Gain*, etc.

Perceptron for Feature Selection Procedure (PFS)

Given a continuous or numeric dataset, with D attributes previously normalized [0,1], and N randomly chosen instances,

1. Let $Acc(t) = 0$ (generalization accuracy), WithoutImprove = ni (number of accepted epochs without accuracy improvement).
2. While $Acc(t+1)$ better than $Acc(t)$ (ni times).
   a. Train a "relaxed" Perceptron (initial weights in zero)
   b. Test after each epoch, and obtain $Acc(t+1)$
   c. If $Acc(t+1)$ better than $Acc(t)$: save weights and do $Acc(t) = Acc(t+1)$
3. Drop attributes with small absolute inter-connection weights.
4. Use the d remain attributes ($d < D$) to create a model as the predictor with some classifier.

Fig. 1. Perceptron for feature selection procedure.

Then, the objective of this Section is to show the exploratory experimentation realized to verify/ invalidate these assumptions (hypothesis).

We conducted several experiments with 8 real and 11 synthetic datasets to empirically evaluate if PFS can do better in selecting features than other well-known feature selection algorithms, in terms of feature reduction, accuracy and processing time. We choose synthetic datasets in our experiments because the relevant features and its inter-dependencies are known beforehand.

3.1 Details

As first experimentation phase, eight real databases were used. The first one is a database with 24 attributes and 2,770 instances; this database contains information of

Mexican electric billing customers, where we expect to obtain patterns of behavior of illicit customers. The next five datasets are taken form the UCI repository [9] (see Table 2 for details).

Additionally, we experiment with two datasets taken from the NIPS 2003 feature selection challenge[1]. These datasets have very high dimensionality but relatively few instances. Specifically, Madelon database has 500 features and 2,000 instances and Gisette dataset has 5,000 features and 6,000 instances.

On the other hand, we test our proposed method with 11 synthetic dataset. Ten of them with different levels of complexity: to obtain these datasets we use the functions described in [10]. Each of the datasets has nine attributes (1.salary, 2.commission, 3.age, 4.elevel, 5.car, 6.zipcode, 7.hvalue, 8.hyears, and 9.loan) plus the class attribute (with class label Group "A" or "B"); each dataset has 10,000 instances. The values of the features of each instance were generated randomly according to the distributions described in [10]. For each instance, a class label was determined according to the rules that define the functions. We experiment also with the corrAL synthetic dataset [11], that has four relevant attributes (A0, A1, B0, B1), one irrelevant (I) and one redundant (R); the class attribute is defined by the function $Y = (A0 \wedge A1) \vee (B0 \wedge B1)$.

In order to compare the results obtained with PFS, we use Weka´s [12] implementation of ReliefF, OneR, CFS and ChiSquared feature selection algorithms. These implementations were run using Weka´s default values, except for ReliefF, where we define 5 as the neighborhood number, for a more efficient response time. Additionally, we experiment with several Elvira´s [13] filter-ranking methods (see Table 1 for details).

To select the best ranking attributes, we use a threshold defined by the largest *gap* between two consecutive ranked attributes, according to [11] (e.g., a *gap* greater than the average *gap* among all the *gaps*). As inductor we utilized Weka´s J4.8 classifier (J4.8 is the last version of C4.5, which is one of the best-known induction algorithms used in data mining) with 10-fold cross validation (although we experimented using more classifiers, we obtained similar results).

In the case of PFS (codified with C language), we set the learning rate α to 0.6, the maximum epochs equal to 500, and the number of epochs without accuracy improvement *ni* to 15, for all the experiments. All the experiments were executed in a personal computer with a Pentium 4 processor, 1.5 GHz, and 250 Mbytes in RAM. In the following sub-Section the obtained results are shown.

3.2 Experiments with 8 real databases

Testing over the Electric billing database, we use the selected features for each method as input to the decision tree induction algorithm J4.8 included in the Weka tool. We notice that PFS obtains similar accuracy as Kullback-Leibler-2, but with less processing time (Table 1). In Fig. 2 we depict a detail of the interconnection weights obtained by each attribute when we apply PFS to the Electric billing database. Only for this database, attributes 22, 23 and 24 are random attributes and, following [14], we use this information to mark the threshold between relevant and irrelevant attributes.

---

[1] http://www.nipsfsc.ecs.soton.ac.uk/datasets/

Table 1. J4.8's accuracies with features selected by each method (Electric billing database).

Method	Total features selected	Accuracy (%)	Pre-processing time
Kullback-Leibler 2	9	97.50	6 secs.
PFS	11	97.29	3 secs.
All attributes	24	97.25	0 secs.
ChiSquared	20	97.18	9 secs.
OneR	9	95.95	41 secs.
ReliefF	4	93.89	14.3 mins.
Euclidean distance	4	93.89	5 secs.
Shannon entropy	18	93.71	4 secs.
Bhattacharyya	3	90.21	6 secs.
Matusita distance	3	90.21	5 secs.
CFS	1	90.18	9 secs.
Kullback-Leibler 1	4	90.10	6 secs.
Mutual Information	4	90.10	4 secs.

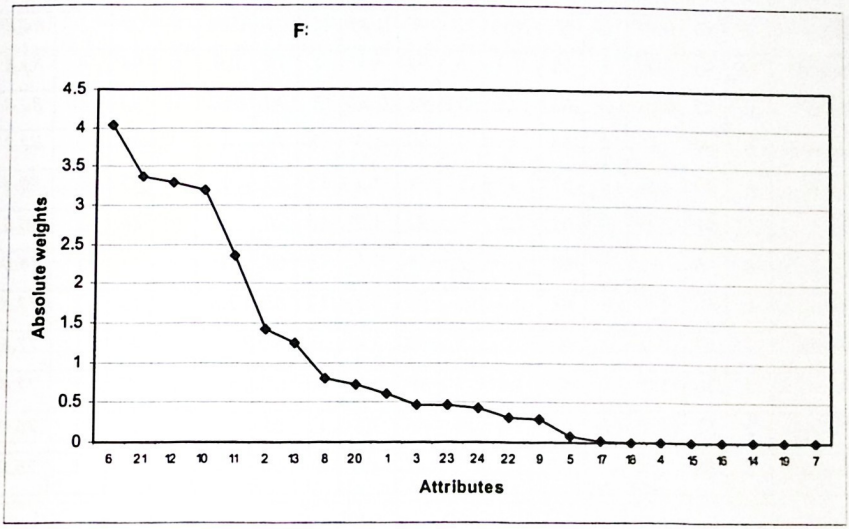


Fig. 2. PFS's weights for each attribute (Electric billing database).

Testing over the five UCI datasets, PFS obtains similar average accuracy as CFS and ReliefF, but with less processing time and good feature reduction (Table 2).

In order to compare our results with real very large databases, we experiment with Madelon and Gisette NIPS 2003 challenge data. In these cases we can not apply Weka or Elvira feature selection tools because they ran out of memory; so, for comparison, we use the results presented by Chen et.al [7]: they apply SVM with a radial basis

function kernel as feature selection method. Table 3 shows results for Madelon and Gisette datasets (N/A means information not mention in [7]).

From Table 3 we can observe that the obtained *BER* using PFS is similar when SVM is applied; on the other hand both, accuracy and *BER*, are poor. The reason for this bad result is because Madelon is a dataset with clusters placed on the summits of a five dimensional hypercube, so, in some sense, is a variation of the XOR problem, a non-linear separable classification problem. Thus, PFS and SVM (still with a kernel function) fail with this database. In the case of Gisette, that contains instances of handwritten digits "4" and "9"; from Table 3 we can see that SVM obtains a superior *BER*, but PFS achieves an acceptable *BER* and accuracy, using few attributes (64 against 913).

Table 2. J4.8's accuracies using the features selected by each method for five UCI datasets.

Method	Vote (16/435)			Horse-c (27/368)			Sick (29/3772)			Sonar (60/208)			Ionosphere (34/351)			Avg. Acc
	TF	Ac	Pt	TF	Ac	Pt	TF	Ac	Pt	TF	Ac	Pt	TF	Ac	Pt	
All atts	16	*96*	0	27	*66*	0	29	*98*	0	60	*74*	0	34	*91*	0	***85.0***
PFS	2	*95*	0.08	3	*68*	0.09	3	*96*	0.4	4	*72*	0.1	5	*93*	0.1	***84.8***
CFS	1	*95*	0.04	2	*66*	0.04	2	*96*	0.25	18	*74*	0.09	8	*90*	3	***84.2***
ReliefF	15	*96*	0.6	3	*66*	0.9	6	*94*	94	4	*70*	0.9	6	*93*	4	***83.8***
SOAP[2]	2	*95*	0.09	3	*66*	0.02	2	*93*	0.12	3	*70*	0.02	31	*90*	0.01	***82.8***
Mutual I	8	*94*	1	4	*68*	1	2	*90*	1.4	18	*73*	1	3	*86*	1	***82.2***
OneR	8	*90*	0.9	3	*67*	1	3	*88*	1.3	12	*72*	1	4	*85*	1	***80.4***
KL-1	5	*91*	1.1	4	*61*	1.2	3	*92*	1.7	16	*70*	1	2	*86*	1	***80.0***
KL-2	4	*88*	1	4	*62*	1.1	2	*89*	1.5	11	*68*	1	3	*83*	1	***78.0***
Matusita	6	*86*	1.9	3	*61*	2.3	2	*91*	3.3	17	*68*	2.5	2	*83*	2	***77.8***
Bhattac	7	*87*	0.9	3	*60*	1	1	*90*	1.4	9	*68*	1	2	*83*	1	***77.6***
Euclidea	8	*86*	1.2	3	*62*	1.4	2	*90*	1.2	10	*67*	1.1	2	*82*	1	***77.4***
ChiSqua	3	*87*	1.4	2	*60*	1.6	3	*88*	1.3	11	*65*	1.2	2	*80*	1	***76.0***
Shannon	3	*86*	1	4	*61*	1.3	2	*87*	1.6	9	*66*	1	2	*80*	1	***76.0***

"(16/435)" means (attributes / instances) for Vote dataset, and so on.
TF=Total features selected Ac=Accuracy (%) Pt=Pre-processing time (secs.)

[2] SOAP's results were taken from [17].

Table 3. Accuracies and *BER* with features selected by each method (Madelon-Gisette).

Database	Method	Features Total (%)	Accuracy (%)	*BER*	Pre-process. time
Madelon	PFS	21 (4. 2%)	58.35	0.4165	48 secs.
	SVM	13 (2. 6%)	N/A	0.4017	N/A
Gisette	PFS	64 (1. 3%)	94. 5	0.0549	3. 3 mins.
	SVM	913 (18. 2%)	N/A	0.0210	N/A

3.3 Experiments with 11 synthetic datasets

The results of applying PFS to 10 synthetic datasets are shown in Table 4. We can observe that the average processing time (column 2) and epochs (column 3) is acceptable. The generalized accuracy obtained for PFS is bad (column 4) but the resulting average accuracy of applies the selected features by PFS to the J4.8 classifier is good (column 5). In columns 6 and 7 we can see that the features selected by PFS are equal or near to the perfect attributes (Oracle column), in almost all cases, except for datasets 3 and 5; the average number of features selected is similar (2.7 vs. 3).

Table 4. PFS with 10 Synthetic Databases.

Database	PFS time (secs)	PFS Epoch	PFS Acc (%)	PFS+J4.8 Acc (%)	PFS Attributes Selected	Oracle
1	3	40	47	100	3-7	3
2	2	24	55	100	1-2-3	1-3
3	2	18	61	68	4	3-4
4	2	17	63	84	1-3	1-3-4
5	3	34	65	82	9	1-3-9
6	4	47	66	99	1-2-3	1-2-3
7	6	59	100	98	9-1-2	1-2-9
8	4	39	100	100	1-2-4	1-2-4
9	4	48	100	97	9-1-2-4	1-2-4-9
10	3	37	99	99	4-8-7-1-2	1-2-4-7-8-9
Avg.	**3.3**	**36.3**	**75.6**	**92.7**	**(2.7)**	**(3)**

Next, we use the selected features obtained by several feature selection methods as input to the decision tree induction algorithm J4.8 (see Table 5).

Table 5. J4.8's accuracies (%) with features selected by each method (10 Synthetic DBs).

Synthetic Database	Method											
	Oracle/All	ReliefF	**PFS**	ChiSquar	Bhattach	Mut.Infor	Kullback Leibler-1	Matusita	OneR	Kullback Leibler-2	Euclidean	Shannon
1	100	100	100	100	100	100	100	100	100	67	100	67
2	100	100	100	73	73	73	73	73	73	73	73	100
3	100	100	68	100	100	100	100	100	100	100	68	59
4	100	90	84	84	84	84	84	84	84	84	84	84
5	100	100	82	74	74	82	74	74	74	82	74	60
6	99	99	99	99	99	99	87	87	99	68	64	69
7	98	98	98	98	94	86	98	86	86	86	88	94
8	100	100	100	100	99	99	100	99	-	99	100	98
9	97	94	97	97	92	85	85	92	85	85	88	85
10	99	80	99	99	97	97	99	97	98	97	97	80
Avg.	99.3	96.1	**92.7**	92.4	91.2	90.5	89.8	89.2	84.9	84.1	83.6	79.6

We use 10-fold cross validation in order to obtain the average test accuracy for each feature subset (in all cases, we obtain similar results using *BER* as quality measure criterion). The column "Oracle/All", from Table 5, represents a perfect feature selection method (it selects exactly the same features that each dataset function uses to generate the class label and, in this case, is equal to the obtained accuracy if we use all the attributes). For dataset 8, only OneR cannot determine any feature subset, because ranks all attributes equally. From Table 5 we can see that the PFS average accuracy is better than several feature selection methods, while worse than only ReliefF.

With respect to the processing time, this is shown in Table 6. We observe that, although PFS is computationally more expensive than ChiSquared and other filter-ranking Elvira's methods, these algorithms cannot detect good relevant attributes or some attribute inter-dependencies; on the other hand, PFS was faster than ReliefF, maintained good generalized accuracy. To have a better idea of the PFS performance, we can compare the results presented previously against the results produced by an exhaustive wrapper approach. In this case, we can calculate that, if the average time required to obtain a classification tree using J4.8 is 1.1 seconds, and if we multiply this by all the possible attribute combinations, then we will obtain that 12.5 days, theoretically, would be required to conclude such a process.

Table 6. Average processing time for each method in seconds (10 Synthetic Datasets)

Exhaustive Wrapper	ReliefF	OneR	**PFS**	ChiSquared and Elvira
1,085,049 (12.5 days)	573 (9.55 mins.)	8	3.3	1

When we test with the corrAL synthetic dataset, PFS was the only that can remove the redundant attribute (Table 7); results for FCBF and Focus methods was taken from [11]. Because the corrAL is a small dataset, processing time in all cases is near to zero seconds, and thus omitted.

Table 7. Features selected by different methods (corrAL dataset).

Method	Features selected
PFS	A0, A1,B0, B1
ReliefF	R, A0, A1, B0, B1
OneR	R, A1, A0, B0, B1
ChiSquared	R, A1, A0, B0, B1
Symmetrical Uncertainty	R, A1, A0, B0, B1
$FCFB_{(log)}$	R, A0
$FCFB_{(0)}$	R, A0, A1, B0, B1
CFS	A0,A1,B0,B1,R
Gain Ratio (Weka)	R, A1, A0, B0, B1
Focus	R

4 Discussion and Related Work

A feature selection line of research is using the scale factors produced by, for example, the Principal Component Analysis technique (PCA), the Support Vector Machine (SVM) variants for Feature Selection (SVM-FS) or the Neural Network (NN) paradigms. These approaches all eliminate the attributes whose associated scale factors are close to zero.

Specifically, in the area of NN, there are several methods for post learning pruning inter-connection weights, for example Optimal Brain Damage or Optimal Brain Surgeon [8]. The objective of these approaches is to obtain a simplified NN, conserving good or similar classification power of the complete NN, and therefore, is not directly focused on the feature selection task. Brank et.al. [15] conducted a study to observe how several scale factor feature selection methods interact with several classification algorithms; however, in this research, no information about processing time and feature reduction is presented.

Ruck et.al. [16] apply feature selection using a Multi-layer Perceptron which is much more complex than a Perceptron; they test with gradient descendent and extended Kalman filtering; again, in this work no information about processing time and feature reduction is presented, and they experiment with few and small datasets.

SOAP [17] is a method that operates on numerical attributes and discrete class and has a low computational cost: it counts the number of times the class value changes with respect to an attribute whose values have been sorted into ascending order. SOAP reduces the number of attributes as compared to other methods; nevertheless, the user has to supply the number of attributes that will be used in the final subset. This is a common problem with the *filter-ranking* methods, that output a ordered list of all attributes, according to its relevance.

5 Conclusions and Future Work

We have presented an easy to implement algorithm for feature selection called PFS that is good trade-off among generalization accuracy, processing time and feature reduction. To validate the algorithm we used 8 real databases and 11 synthetic datasets.

The results show that PFS represent a good alternative, simple but effective, compared to other methods, because its acceptable processing time, accuracy and good performance in the feature selection task (feature selection ratio). For the case of the real Electric billing database, PFS obtains similar, or better, accuracy and feature reduction as Kullback-Leibler-2, CFS or ReliefF, but with the half of processing time. Similar comments apply for the rest of the experimented datasets.

The proposed PFS algorithm has several advantages: (1) it requires a linear amount of memory; (2) its generalization accuracy and processing time is competitive against other methods, although its time complexity is a function of the epochs without accuracy improvement (Perceptron´s convergence); (3) does not realize exhaustive or combinatorial search; (4) finds some attribute inter-dependencies; and (5) obtains acceptable feature reductions. The main disadvantage is its limitation to classify only linear separable datasets.

Some future works arise with respect to PFS improvement. For example: apply kernel functions to overcome the linear separability class limitation; try with other learning stopping criteria; realize experiments using a metric (e.g., *F-score*) to do a first attribute elimination, and then apply PFS. Another future work will be the application of the formalism to other very large power system databases such as the national power generation performance database, the national energy control databases, the energy de-regulated market and the Mexican electric energy distribution database.

References

1. Guyon, I., Elisseeff, A., An introduction to variable and feature selection, Journal of machine learning research, 3 (2003) 1157-1182.
2. Piramuthu, S., Evaluating feature selection methods for learning in data mining applications, Proc. 31st annual Hawaii Int. conf. on system sciences (1998) 294-301.
3. Molina, L., Belanche, L., Nebot, A., FS algorithms, a survey and experimental evaluation, IEEE Int.conf.data mining, Maebashi City Japan (2002) 306-313.
4. Mitra, S., et.al., Data mining in soft computing framework: a survey, IEEE Trans. on neural networks, vol. 13, no. 1, January (2002) 3-14.
5. Kohavi, R., John, G., Wrappers for feature subset selection, Artificial Intelligence Journal, Special issue on relevance (1997) 273-324.
6. Gallant, S.I.: Perceptron-Based Learning Algorithms, in IEEE Transactions on Neural Networks, 1 (1990) 179-191.
7. Chen,Y., Lin,C., Combining SVMs with various feature selection strategies. To appear in the book "Feature extraction, foundations and applications", Guyon, I. (ed) (2005).
8. Jutten, C., Fambon, O., Pruning methods: a review, European symposium on artificial neural networks, April (1995) 129-140.
9. Newman, D.J. & Hettich, S. & Blake, C.L. & Merz, C.J. UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science (1998).

10. Agrawal, R., Imielinski, T, Swami, A., Database mining: a performance perspective, IEEE Trans. Knowledge data engrg. Vol. 5, no. 6 (1993) 914-925.
11. Yu, L., Liu, H., Efficient feature selection via analysis of relevance and redundancy, Journal of Machine Learning Research 5 (2004) 1205-1224.
12. www. cs.waikato.ac.nz/ml/weak (2004).
13. www. ia.uned.es/~elvira/ (2004).
14. Stoppiglia, H., Dreyfus, G., et.al., Ranking a random feature for variable and feature selection, Journal of machine learning research, 3 (2003) 1399-1414.
15. Brank, J., Grobelnik, M., Milic-Frayling, N. & Mladenic, D., Interaction of feature selection methods and linear classification models. Proceedings of the ICML-02 Workshop on Text Learning, Sydney, AU (2002).
16. Ruck, D.W., Rogers, S.K., Kabrisky, M. Feature selection using a multilayer perceptron. Journal of Neural Network Computing, 2 (1990) 40-48.
17. Ruiz, R., Aguilar, J., Riquelme, J., SOAP: efficient feature selection of numeric attributes, VIII Iberamia, workshop de minería de datos y aprendizaje, Spain (2002) 233-242.

A Modified Bottleneck Neural Network for Dimensionality Reduction

Eduardo Filemón Vázquez Santacruz, Debrup Chakraborty

Centro de Investigaciones y Estudios Avanzados del IPN
Av. IPN 2508, Col: San Pedro Zacatenco
Mexico D.F, Mexico
email:evazquez@computacion.cs.cinvestav.mx, debrup@cs.cinvestav.mx

(Paper received on July 10, 2007, accepted on September 1, 2007)

Abstract. We present a modification of the bottleneck neural network for dimensionality reduction. We call our scheme a modified bottleneck network (MBNN). Unlike a traditional bottleneck network for dimensionality reduction, an MBNN uses the class information and thus the transformed data can be suitably used for classification problem. We also propose a new technique to create ensembles of neural networks using multiple projections of the same data obtained from different MBNNs. We justify the suitability of the proposed method by some experiments on some classification problems.

1 Introduction

Dimensionality reduction is probably one of the most important task in any pattern recognition problem. This problem has been addressed from different viewpoints, and by various kinds of tools. Naively, dimensionality reduction can be defined as follows. Given the input data $X = \{x : x \in \Re^p\}$, we want to find a transformation $\Phi : X \rightarrow X'$, where $X' = \{x' : x' \in \Re^q\}$ and $q < p$. Typically the set X' optimizes certain criteria J. The criteria J usually depends on the problem that is being solved, like for a classification problem J may be a measure of class separability or the misclassification rate for a fixed classifier. Two important types of dimensionality reduction are feature selection, where only a subset of the p features present in the data points in X are selected and an other type is feature extraction where the data is projected into some low dimensional space. In the second type of transformation, the original features present in the data may loose their individual identity and the final features obtained in X' may be a combination of all features present in the data points of X. Note that the term feature extraction is a generic term which signifies extracting features from a given set of features and includes such transforms which increases the dimensionality of a given data set.

Dimensionality reduction enables obtaining a data whose representation is less complex. This data thus can be learned by a learning machine which is less complex. This gives rise to a computational saving in construction and use of the learning machine. Also dimensionality reduction can give rise to improved prediction accuracies in scenarios where the training sample size is small (which is true in most real life scenarios)[12].

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 127-136

The problem of dimensionality reduction has been well addressed in literature and it has been tried out in various paradigms. Previous studies on dimensionality reduction focused mainly around statistical approaches like Principal Components Analysis (PCA)[14], linear discriminant analysis (LDA) [10] etc. These methods attempt to reduce the dimensionality of the feature space by creating new features which are combination of the original ones. Hence, PCA and related methods are feature extraction techniques which extract a new set of features from the available set of features, and the dimensionality of the extracted feature space is less than that of the original one. There are numerous variants of the PCA which tries to solve the projection problem using various modifications of the main PCA technique [24]. The main drawback of these methods is that the new features lose their original identity. Leaving aside the feature extraction methods, there have been other works on feature selection using statistical techniques [13, 18].

Blum and Langley [4] have given an excellent survey for selection of relevant features in machine learning. These approaches are different in evaluation of the feature subsets. One can broadly classify the approaches as filter approaches and wrapper approaches. In filter approach, the feature evaluation index is independent of the main classification/function approximation algorithm, whereas in wrapper approaches the features are evaluated by the main algorithm itself. Wrapper approaches are considered better as the relevance of a feature is generally dependent on the task being performed and also on the tool being used to do the task [16].

There are many feature selection algorithms that use soft computing or computational intelligence tools. Methods described in [6, 19] use genetic algorithms to select the relevant feature subsets. Methods described in [20, 22, 23, 25] and a variety of others use neural networks for feature selection. Feature selection has also been attempted using fuzzy and neuro-fuzzy techniques [8, 21].

Our contribution: In this paper we discuss a technique to do dimensionality reduction using a neural network. Our method is a feature extraction technique. Our technique projects a given data into a low dimensional space while preserving the class separability of the data. Our technique depends on an special neural architecture called the bottleneck neural network. The bottleneck neural network has been previously used for data compression. The traditional bottleneck network do not use class information present in a data thus the compression achieved is sort of unsupervised, which may not be suited for classification problems. We discuss a simple variant of the bottleneck network which uses the class information for classification. We discuss several ways to use the transformed data for classification. We also propose a new technique to create neural network ensembles using the projected data obtained from a modified bottleneck network. We test the method on some real life classification problems.

2 The Bottleneck Neural Network

The bottleneck neural network is a nice strategy which has been used for data compression. Given a data set $\Xi = \{\xi_1, \xi_2, \ldots \xi_m\} \subset \Re^p$, a multilayered perceptron

(MLP) with p nodes in the input and output layer and q ($q < p$) nodes in a hidden layer is trained. Further we shall refer to such an architecture with the notation $p : q : p$. For training this network input-output pairs of the form (ξ_i, ξ_i) $(i = 1, 2, \ldots n)$ are used. So the network is trained to learn an identity map, and for a properly trained network, the output for ξ_i would be ξ_i itself. Now, the output of the hidden layer for each ξ_i would be a q dimensional representation of ξ_i. Thus a reduction of dimensionality is obtained. The bottleneck neural network has been previously used in many applications [3, 17].

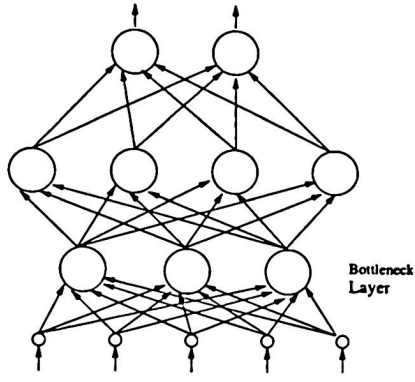


Fig. 1. The Modified Bottleneck Network

2.1 Modified Bottleneck Network

Here we propose a variant of the bottleneck neural network which we call the modified bottleneck network (MBNN). Let $X = \{(x_i, y_i) : i = 1, 2, \ldots n\}$ be a training data where $x_i \in \Re^p$ and $y_i \in \Re^s$. We train an MLP with this data. We have no restriction on the architecture of the MLP except that at least one of the hidden layers (say, the r^{th} layer) contains q (where $q < p$) nodes, we call this layer as the bottleneck layer. Note, unlike the traditional bottleneck network in MBNN we are not training the network to learn an identity map but our network tries to learn the real input-output mapping present in the data.

Let $\mathcal{B}$ denote a MBNN. $\mathcal{B}$ is not different from an ordinary MLP in terms of structure or the learning algorithm except the minimal restriction on its architecture as stated earlier. The output of $\mathcal{B}$ is not the output of its output layer but is the output of the bottleneck layer. For example, let us consider a data set $S = \{(x, y) : x \in \Re^5, y \in \Re^2)\}$. Figure 1 shows a MBNN trained with S. As evident from Fig. 1, the MBNN has an architecture of $5 : 3 : 4 : 2$ with the first hidden layer (the hidden

layer containing 3 nodes) as the bottleneck layer. One could have also chosen the second hidden layer as the bottleneck layer. The network is trained with the data set S using any convenient learning algorithm like the backpropagation or some of its variants. For using this network, the output is tapped from the first hidden layer. So for any input vector $\xi \in \Re^5$, the output of the network would be $\alpha \in \Re^3$.

For the general case, let $\mathcal{B}_X$ be a MBNN trained with $X = \{(x_i, y_i) : x \in \Re^p, y \in \Re^s)\}$. Let the r^{th} layer of $\mathcal{B}_X$ be the bottleneck layer with q nodes. Let $\alpha_i \in \Re^q$ denote the output of the r^{th} layer of $\mathcal{B}_X$ for a data point x. From a properly trained neural network we collect the output of the r^{th} node and this output serves as a projection of the original p dimensional data in a q dimensional space. We call this data as the *reduced data*. It is likely that the reduced data retains the properties required for learning the underlying function that the original data represents. This is because the reduced data represents an internal configuration of the data within the network, and truly this configuration helps the neural network in learning the proper mapping present in the data.

The reduced data set obtained from a trained network can further be used to train a new network possibly with a smaller architecture. Thus this will lead to lesser training time and possibly better generalization. Next we discuss some of the ways in which the reduced data obtained from an MBNN can be used.

3 How to Use the Reduced Data

An MBNN transforms a data set into a low dimensional space using a nonlinear transform. Also, an important feature of this transform is that it uses the output information (eg. the class labels for a classification data) for performing the transform. The reduced data represents an internal configuration of a trained network which further gets transformed into the output. This intermediate configuration is thus suitable to be used in any other neural network. Next we discuss some scenarios where a reduced data set may be useful.

3.1 As a Transform Before Training an MLP for Prediction

It is common knowledge that as dimensionality of a data goes up we need a bigger network to learn the data. A bigger network means more adjustable parameters, and as the number of parameters increases, the possibility of a network to overfit becomes more. An overfitted network though performs good with the training data can miserably fail to give acceptable results on test data sets. This problem becomes more acute if the number of training samples are small. To avoid this a natural and popular practice is to apply a dimensionality reduction on the data before training a network with the data. Thus by reducing the dimension of the original data, the data can be learned by a smaller network and thus possibility of overfitting becomes less. The most used technique is to use Principal Components Analysis (PCA) on the data. The PCA technique is a generic technique for dimensionality reduction and it projects the data in a new space where it has maximum variance. This may

not be the best objective for projection when we know what we need to do with the data. For example, when the desired task is classification, class separability may be a better objective than maximizing the variance. Also, if the targeted tool is a multilayered perceptron, which functions in a highly nonlinear manner and produces highly nonlinear class boundaries, use of the PCA (or other related projection techniques like LDA) may not be always useful. The reduced data set obtained by a MBNN will have better representation than the other known data projection methods in terms of classifiability by a neural network. The reason for this being that the reduced data obtained by a MBNN is an internal representation of a neural network meant for classifying the data.

A critique to this method may be that we need to train a network to get the reduced data itself. Thus for data sets with huge dimensions the MBNN may itself overfit the data if we need a network which performs good on the training data. This is of course true. But as the MBNN is not used for prediction purposes, an overfitted MBNN may not do us much harm. And we extract the reduced data of low dimension from an MBNN and train a new network (possibly much smaller than the MBNN itself), thus the network we use for prediction will have better generalization capabilities.

3.2 In Constructing Neural Network Ensembles

Ensemble methods like bagging [5] and boosting [9] can enhance the prediction ability of any classifier to a great extent. Neural network ensembles has also been reported to perform better than single networks. Bagging is a method very suited to neural networks, as bagging can decrease the variance of predictions in classifiers, and neural networks which are known to be "unstable" have large variance in prediction [5]. For neural networks, bagging involves creation of multiple bootstrap samples from a given data set and training multiple neural networks with those bootstrap samples. The final result is obtained by suitably aggregating the outputs of the candidate networks of the ensemble. Thus creating a bagging ensemble of neural networks is quite expensive in terms of computation, as neural network training is expensive. Using an bottleneck neural network $\mathcal{BN}$ one can project the data to a low dimensional space, and use bootstrap samples of the projected data to train individual candidates of the ensemble. This can give considerable savings in training times.

3.3 In Methods Which Uses Explicit or Implicit Density Estimation

There is a family of works which observes that one of the reasons of the phenomenon of overfitting in MLPs is due to the fact that MLPs are trained with a finite training sample [7, 11, 15]. In a conventional training algorithm for training MLPs, the network gets trained with a fixed sample of training points in each epoch. In [11] it was proposed that overfitting can be reduced if the training set can be indefinitely expanded, and the network be trained with a new set of data in each epoch. To

achieve this, the authors in [11], use a method to estimate the probability density of the data, and generate a new data set following the same probability density in each epoch, thus the MLP faces new points in each epoch. In [15], the method reported in [11] was modified by improving the procedure for density estimation. In [7] another method to expand the training set was reported and it do not depend on explicit density estimation but uses a k-NN heuristic to generate new points in each epoch.

All these methods reported in [7, 11, 15] can reduce the overfitting in MLP networks to a great extent. But as in [11, 15] a sort of a kernel density estimate is considered using gaussian kernels, the correctness of the density estimate decreases with the increase of dimension of the data. In [7] a K-nn technique is used which also fails for high dimensional data. An easy fix of this problem may be to use the reduced data obtained from an MBNN for training and density estimation not the original data set.

3.4 Bagging with Multiple Projections of the Same Data

As discussed earlier bagging involves aggregating outputs of k classifiers trained with k different bootstrap samples drawn from a given data set X. The idea behind bagging is to create independent classifiers from the same data set. Each bootstrap samples thus acts as a different data set for each classifier. Here we propose to use different projections of the same data to train each candidate of the ensemble. We propose to use the MBNN for projecting the data.

Given a data set $X = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) : i = 1, 2, \ldots n, \boldsymbol{x} \in \Re^p, \boldsymbol{y} \in \Re^s\}$ we train k MBNNs $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k$. Each $\mathcal{B}_i$ can have different architectures with different hidden layers and different hidden nodes. The only restriction in each $\mathcal{B}_i$ being that the input layer should contain p nodes, the output layer should contain s nodes, and for any hidden layer should contain q(where $q < p$) nodes. From each of the networks we can collect a different projection of the original data X, possibly in different dimensions. Thus using each $\mathcal{B}_i$ we obtain the following transform $\mathcal{B}_i : X \rightarrow X_i'$. Thus we train k new neural networks using the data sets X_i' for $i = 1, \ldots, k$. We create an ensemble of these k networks, and aggregate the final outputs using a suitable aggregation function.

4 Experimental Results

We study the suitability of this method by applying it to some real life classification problems. The data sets used are Lung Cancer, Sonar, Iono, DNA and Protein. The summary of the data sets are presented in Table 1.

In all the simulations performed we used the neural network tool-box of MATLAB. We used "traingdx" as the learning algorithm, which is a variant of backpropagation with an ability to automatically tune the learning rate when necessary. Tables 2,3,4,5 and 6 summarizes the results. Each row in the tables stands for a different method and the columns report the average performance (with standard deviation) in percentage on the test data. Each reported result is of a 10 fold cross

Table 1. Summary of the data sets

Data	Source	Number of points	No of features	No of Classes
Lung Cancer	[2]	32	56	3
Sonar	[2]	208	60	2
Iono	[2]	351	34	2
DNA	[2]	2000	180	3
Protein	[1]	698	125	4

validation repeated 20 times. We explain each row of each table in more details as follows.

Table 2. Results on Lung Cancer

	Methods	Performance
1	MBNN Single	80.16667 ± 4.1578
2	MBNN Bagging	80.6667 ± 5.23
3	MBNN MP Bagging	83.7778 ± 3.674
4	PCA Single	78.1667 ± 3.6388
5	PCA Bagging	75.8333 ± 4.2492
6	Full Data Single NN	77.1667 ± 5.6501
7	Full Data Bagging	82.44 ± 2.194

1. **MBNN Single:**We trained a MBNN with 10 nodes in the hidden layer to get a 10 dimensional reduced data. We trained 20 different networks with the same reduced data each having the same architecture with 10 nodes in a single hidden layer. Row 1 of the tables shows the average performance of those 20 networks.
2. **MBNN Bagging** Next using the same reduced data of 10 dimension as obtained in row one we perform a bagging with 10 candidates in the ensemble. We use the majority vote to aggregate the results of the 10 different candidate networks. We created 20 different ensembles using the same reduced data. Row 2 shows the average performance of those 20 ensembles.
3. **MBNN MP Bagging:** MP bagging signifies bagging with multiple projections. In this experiment we trained 10 different MBNN each with 10 nodes in the bottleneck layer and thus obtained 10 different projections of the data in 10 dimensions. We trained 10 different networks with 10 nodes in a single hidden layer and aggregated their results using majority voting. This experiment was also performed 20 times. The results of this experiment are summarized in row 3 of the tables.
4. **PCA Single:**We ran a Principal Components Analysis on the original data with all features and took the 10 most significant components thus reducing the

Table 3. Results on Sonar

	Methods	Performance
1	MBNN Single	82.1143 ± 3.048
2	MBNN Bagging	82.1929 ± 3.2514
3	MBNN MP Bagging	85.7929 ± 0.6167
4	PCA Single	77.6786 ± 1.4881
5	PCA Bagging	78.7857 ± 1.1756
6	Full Data Single NN	79.6214 ± 2.8686
7	Full Data Bagging	84.5429 ± 0.8891

Table 4. Results on Iono

	Methods	Performance
1	MBNN Single	89.9786 ± 0.9993
2	MBNN Bagging	90.2937 ± 1.1636
3	MBNN MP Bagging	90.8341 ± 0.2622
4	PCA Single	86.7246 ± 0.7528
5	PCA Bagging	86.1579 ± 0.6861
6	Full Data Single NN	87.8389 ± 1.7492
7	Full Data Bagging	91.0611 ± 0.9597

Table 5. Results on DNA

	Methods	Performance
1	MBNN Single	92.425 ± 0.675
2	MBNN Bagging	92.49 ± 0.5611
3	MBNN MP Bagging	94.585 ± 0.094428
4	PCA Single	90.72 ± 0.3048
5	PCA Bagging	90.6 ± 0.2856
6	Full Data Single NN	89.3 ± 0.1768
7	Full Data Bagging	94.8 ± 0.2619

Table 6. Results on Protein

	Methods	Performance
1	MBNN Single	76.3377 ± 1.4511
2	MBNN Bagging	76.4 ± 1.7995
3	MBNN MP Bagging	79.5584 ± 0.459
4	PCA Single	63.7143 ± 1.2786
5	PCA Bagging	64.4 ± 1.3853
6	Full Data Single NN	69.7633 ± 4.1204
7	Full Data Bagging	78 ± 0.5898

original data to a 10 dimensional data. Row 4 of the tables shows the average performance and the best performance among 20 MLPs with 10 hidden nodes in a single hidden layer trained with the data of 10 features obtained by using PCA.

5. **PCA Bagging:** The 10 dimensional data obtained by PCA is also used to train an ensemble of 10 MLPs (each 10 nodes in a single hidden layer) using bagging. Row 5 of the tables gives the average and best performance of 20 such ensembles created with the 10 dimensional data obtained by PCA.
6. **Full Data Single NN** Row 6 gives the result on the data with all features using a ordinary MLP.
7. **Full Data Bagging** Row 7 gives the results on the data sets for bagging using the full dimensional data.

The results clearly show that the reduced data retains the class separability for all data sets experimented with. For all the data sets the MBNN single gives a better result than a single neural network trained with the whole data. The results obtained by using a PCA for dimensionality reduction are significantly poorer than the results obtained using a MBNN. MBNN MP bagging gives very encouraging results in all cases. In fact the results obtained by MBNN MP bagging outperform all other methods and is quite comparable with the results obtained by bagging on the full data set.

5 Conclusion

We presented a simple modification of the traditional bottleneck network for dimensionality reduction. We also showed some specific ways to use the reduced data obtained from an MBNN. The method of multiple projection bagging seems to work quite well as a method to create neural network ensembles. The methods are tested on some real life classification problems and the results obtained are encouraging.

References

1. http://www.nersc.gov/~cding/protein
2. A. Asuncion and D.J. Newman, UCI Machine Learning Repository [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science
3. Y. Araki, T. Ohki, D. Citterio, M. Hagiwara, K. Suzuki, "A new method for inverting feedforward neural networks," IEEE International Conference on Systems, Man and Cybernetics, 2003. Volume 2, pp.1612 - 1617, 2003
4. A.L. Blum, P. Langley, "Selection of relevant features and examples in machine learning", *Artificial Intelligence*, vol 97, no 1, pp. 245-271, 1997.
5. Leo Breiman, "Bagging predictors", *Machine Learning*, vol. 24, no. 2, pp. 123-140, 1996.
6. F.Z. Brill, D.E. Brown and W.N. Martin, "Fast genetic selection of features for neural network classifiers", *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 324-328, 1992.

7. D. Chakraborty and N. R. Pal, "Expanding the training set for better generalization in MLP," Proceedings of *International Conference on Communication, Devices and Intelligent Systems*, CODIS-2004, pp. 454-457, 2004.
8. R. De, N.R. Pal and S.K. Pal, " Feature analysis: neural network and fuzzy set theoretic approaches", *Pattern Recognition* vol 30, no 10,pp. 1579-1590, 1997.
9. Y. Freund and R.E. Schapire, "A short introduction to boosting", *Journal of Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771-780, 1999.
10. K. Fukunaga, *Statistical Pattern Recognition*, Academic Press, San Diego, CA, USA, 1991.
11. L. Holmstrom and P. Koistinen, "Using additive noise in backpropagation training," *IEEE Trans. Neural Networks*, vol. 3, pp. 24-38, 1992.
12. A. K. Jain, B. Chandrasekaran, "Dimensionality and sample size considerations in pattern recognition practice," in P.R. Krishnaiah, N.L. Kanal (eds.), *Handbook of Statistics*, vol 2, North-Holland, Amsterdam, pp. 835-855, 1982.
13. A. K. Jain and D. Zongker, "Feature selection: evaluation, application and small sample performance," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 153-148, 1997
14. I. T. Jollife, *Principal Component Analysis*, Springer Verlag, New York, 1986.
15. G. N. Karystinos and D. A. Pados, "On overfitting, generalization, and randomly expanded training sets," *IEEE Trans Neural Networks* vol 11, no. 5, pp. 1050-1057, 2000.
16. F. Kohavi and G. John, "Wrappers for feature subset selection", *Artificial Intelligence*, vol 97, no 1, pp. 273-342, 1997.
17. M. Marseguerra ans A. Zoia, "The autoassosiative neural network in signal analysis: I. Data dimensionality reduction and its geometric interpretation", *Annals of Neuclear Energy*, vol. 32, pp. 1191-1296, 2005.
18. J. Novovicova, P. Pudil and J. Kittler, "Divergence based feature selection for multimodal class densities", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol 18, no 2, 1996.
19. M.L. Raymer, W.F. Punch, E.D. Goodman, L.A. Kuhn and A.K. Jain, "Dimensionality reduction using genetic algorithms", *IEEE Trans. on Evolutionary Computing*, vol 4, no 2, pp. 164-171, 2000.
20. D.W. Ruck, S.K. Rogers, M. Kabrisky, "Feature selection using a multilayered perceptron", *Journal of Neural Network Computing*, pp. 40-48, 1990.
21. M. R. Rezaee, B. Goedhart, B. P. F. Lelieveldt and J.H.C. Reiber, "Fuzzy feature selection," *Pattern Recognition*, vol. 32, pp. 2011-2019, 1999.
22. R. Setino, "Neural network feature selector", *IEEE Trans. Neural Networks*, vol 8, pp.654-662, 1997.
23. J.M. Steppe Jr, "Integrated feature and architecture selection", *IEEE Trans. Neural Networks*, vol 7, pp. 1007-1014, 1996.
24. M. Ture, I. Kurt, Z. Akturk,"Comparison of dimension reduction methods using patient satisfaction data", *Expert Systems with Applications*, Vol 32, pp. 422-426, 2007
25. J.M. Zurada, A. Malinowski and S. Usui, "Perturbation method for detecting redundant inputs of perceptron networks", *Neurocomputing*, vol 14, pp. 177-193, 1997.

Real World Applications

Multiple Fault Diagnosis in Electrical Power Systems with Dynamic Load Changes using Probabilistic Neural Networks

Juan Pablo Nieto González[1], Luis E. Garza Castañón[2], Rubén Morales Menendez[3]

I.T.E.S.M. Saltillo Campus Mechatronics Department
[1]Prol. Juan de la Barrera No. 1241 Ote. Colonia Cumbres,
25020, Saltillo, Coahuila, México
juan.pablo.nieto@itesm.mx

I.T.E.S.M. Monterrey Campus
[2]Mechatronics and Automation Department, [3]Center for Industrial Automation
Ave. Eugenio Garza Sada No. 2501 Sur, Colonia Tecnológico
64849, Monterrey, Nuevo León, México
legarza@itesm.mx, rmm@itesm.mx

(Paper received on June 22, 2007, accepted on September 1, 2007)

Abstract. Power systems monitoring is particularly challenging due to the presence of dynamic load changes in normal operation mode of network nodes, as well as the presence of both continuous and discrete variables, noisy information and lack or excess of data. This paper proposes a fault diagnosis framework that is able to locate the set of nodes involved in multiple fault events and detects the type of fault in those nodes. The framework is composed of two phases: In the first phase a probabilistic neural network is trained with the eigenvalues of voltage data collected during symmetrical and unsymmetrical fault disturbances. The second phase is a sample magnitude comparison used to detect and locate the presence of a fault. A set of simulations is carried out over an electrical power system to show the performance of the proposed framework and a comparison is made against a diagnostic system based on probabilistic logic.

1 Introduction

As processes become more complex, the monitoring of them is very important in order to improve process performance, efficiency and product quality. Monitoring of industrial processes plays a substantial role in system safety, availability and production quality. Early detection of faults can help to avoid major breakdowns and incidents. In order to tackle those problems, fault detection and system diagnosis has been an active research domain since years.

There exist a lot of research works related with fault detection. Most of the methods used are analytic, based on artificial intelligence (AI) or statistical methods. [1] classifies fault detection and isolation methods in three groups. 1)Quantitative Model Based, 2) Qualitative Model Based and 3) Process History Based.

© H. Sossa, R. Barrón, and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 139-150

Quantitative Model Based fault detection methods are based on a mathematical model of the system. The occurrence of a fault is captured by discrepancies between the observed behavior and the one that is predicted by the model. These approaches make use of state estimation, parameter identification techniques, and parity relations to generate residuals. Fault localization then, rest on interlining the groups of components that are involved in each of the detected discrepancies. However, it is often difficult and time-consuming to develop accurate mathematical models that characterize all the physical phenomena occurring in industrial processes.

Qualitative Model Based fault detection methods use symbolic reasoning which generally combines different kinds of knowledge with graph theory to analyze the relationships between variables of a system. An advantage of these methods is that an explicit model of the system to be diagnosed is not necessary. Knowledge-based approaches such as expert systems may be considered as alternative or complementary approaches where analytical models are not available.

Process History Based fault detection methods only require a big quantity of historical process data. There are several ways in which these data can be transformed and presented as prior knowledge of a system. These transformations are known as feature extraction and could be qualitative, as those used by expert systems, and qualitative trend analysis methods or quantitative, as those used in neural networks, PCA, PLS or statistical pattern recognition.

The reasons behind the increased interest in fault diagnosis in power networks are the complexity and high degree of interconnection present in electrical power networks, that can lead to an overwhelming array of alarms and status messages being generated as a result of a disturbance. This can have a negative impact on the speed with which operators can respond to a contingency. Therefore, in order to increase the efficiency of diagnosis, it is necessary to use automated tools, which could help the operator to speed up the process. Very recently, the need to develop more powerful approaches has been recognized, and hybrid techniques that combine several reasoning methods start to be used [2]. This approach incorporate model based diagnosis and signal analysis with neural networks. [3] presents Bayesian networks (BNs) to estimate the faulty section of a transmission power system. Simplified models of BNs with Noisy-Or and Noisy-And nodes are proposed to test if any transmission line, transformer, or busbar within a blackout area is faulty. In [4] an investigation is performed about the use of logistic regression and neural networks to classify fault causes. This paper also discusses about data insufficiency, imbalanced data constitution and threshold setting. Ren and Mi [5] propose a procedure for power systems fault diagnosis and identification based on Petri Nets and coding theory. They tested the approach with simulations over the IEEE 118-bus power system and highlight the great advantage to handle very easily future expansions. In [6] a Fault diagnosis system is presented, based on multi-agent systems. By using a negotiation mechanism between decision-making agent and a cooperative agent, fault diagnosis results can be obtained.

In this paper it is proposed a multiple fault diagnosis framework composed by two phases. Eigenvalues are computed from the correlation matrix which is built from historical data, and then are used as a probabilistic neural network inputs. In first phase a most probably component state of each node is given and in second phase the comparison of each sample against a constant value gives the real component state and the location of a fault, finally the diagnosis of the system is carried out.

The organization of the paper is as follows: section 2 explains probabilistic neural networks basis and gives the correlation matrix and eigenvalues definitions. Section 3 gives the framework general description. Section 4 shows how the framework works in a simulation example with single and multiple faults as well as a comparison of the general performance of it against a diagnostic system based on probabilistic logic. Section 5 concludes the paper.

2 Preliminary

2.1 Probabilistic Neural Network Basis

PNN are conceptually similar to K-Nearest Neighbor (KNN) models [7]. The basic idea is that a predicted value of an item is likely to be about the same as other items that have close values of the predictor variables.

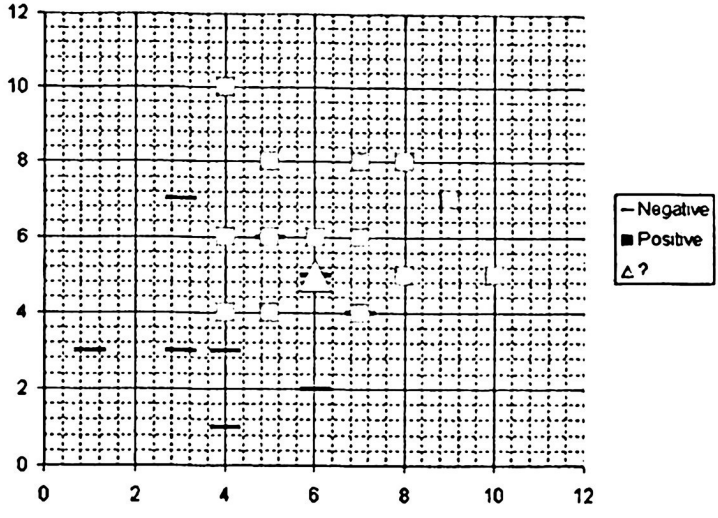


Fig. 1. PNN are conceptually similar to KNN

From Fig. 1 it is assumed that each case in the training set has two predictor variables, x and y. The cases are plotted using their x,y coordinates as shown in the figure. Also it is assumed that the target variable has two categories, positive which is denoted by a square and negative which is denoted by a dash. It can be note that the triangle is positioned almost exactly on top of a dash representing a negative value. But that dash is in a fairly unusual position compared to the other dashes which are clustered below the squares and left of center. So it could be that the underlying negative value is an odd case. The nearest neighbor classification will depend on how many neighboring points are considered. If 1-NN is used and only the closest point is

considered, then the new point should be classified as negative since it is on top of a known negative point. On the other hand, if 9-NN classification is used, the closest 9 points are considered and then the effect of the surrounding 8 positive points may overbalance the close negative point. A probabilistic neural network builds on this foundation and generalizes it to consider all of the other points. The distance is computed from the point being evaluated to each of the other points, and a radial basis function (RBF) (also called a kernel function) is applied to the distance to compute the weight (influence) for each point. The radial basis function is so named because the radius distance is the argument to the function. *Weight=RBF(distance)* the further some other point is from the new point, the less influence it has. Different types of radial basis functions could be used, but the most common is the Gaussian function. The PNN architecture is shown in figure 2. The model has two layers: radial basis layer and competitive layer.

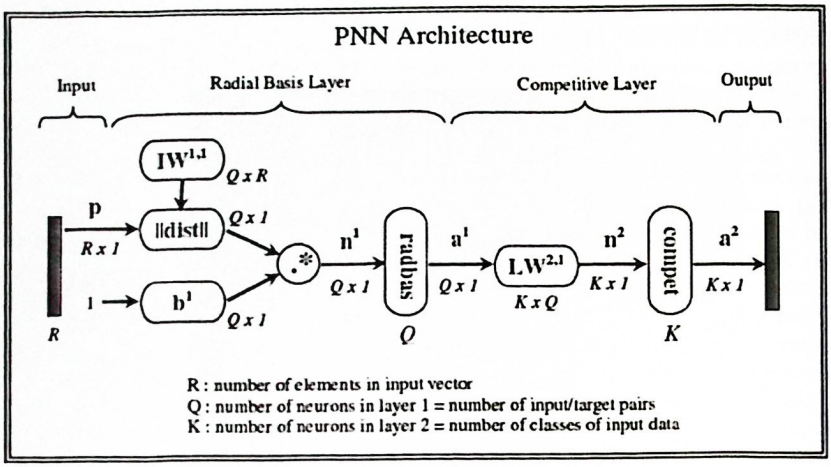


Fig. 2. PNN architecture.

There are Q input vector/target vector pairs. Each target vector has K elements. One of these element is 1 and the rest is 0. Thus, each input vector is associated with one of K classes. When an input is presented the $\|dist\|$ box produces a vector whose elements indicate how close the input is to the vectors of the training set. An input vector close to a training vector is represented by a number close to 1 in the output vector a^1. If an input is close to several training vectors of a single class, it is represented by several elements of a^1 that are close to 1. Each vector has a 1 only in the row associated with that particular class of input, and 0's elsewhere. The multiplication Ta^1 sums the elements of a^1 due to each of the K input classes. Finally, the second layer, produces a 1 corresponding to the largest element of n^2, and 0's elsewhere. Thus, the network has classified the input vector into a specific one of K classes because that class had the maximum probability of being correct.

2.2 Correlation Matrix and Eigenvalues Definitions

Correlation matrix definition. A Correlation matrix describes correlation among M variables. It is a square symmetrical M x M matrix with the (ik)th element equal to the correlation coefficient r_{ik} between the (i)th and the (k)th variable. The correlation coefficient is obtained as

$$r_{ik} = \frac{\sum_{j=1}^{n}(x_{ji} - \bar{x}_i)(x_{jk} - \bar{x}_k)}{\sqrt{\sum_{j=1}^{n}(x_{ji} - \bar{x}_i)^2}\sqrt{\sum_{j=1}^{n}(x_{jk} - \bar{x}_k)^2}} \tag{1}$$

The diagonal elements (correlations of variables with themselves) are always equal to 1.00 [8].

Eigenvalue definition. Let A be a k x k square matrix and I be the k x k identity matrix. Then the scalars

$$\lambda_1, \lambda_2, \ldots, \lambda_k \tag{2}$$

satisfying the polynomial equation

$$|A - \lambda I \tag{3}$$

are called the eigenvalues or characteristic roots of a matrix A. The equation $|A - \lambda I| = 0$ is called the characteristic equation, thus similar matrices and A and its transpose matrix have same eigenvalues [8].

3 Framework Description

The proposed detection framework is shown in figure 3. As the framework is a Process History Based fault detection method, this only requires a big quantity of historical data of the power system's nodes containing normal operation data and faulty data samples from the different types of faults that could be present in the system. These data sets are used as prior knowledge of the power system to perform the detection process.

The first step is to obtain several data sets from the power system's nodes. These data sets are matrices formed by windows of m samples and n power system's nodes where the voltage of each line of a certain node is monitored, that means three readings per node. Such matrices are constructed with normal node operation and different node faults present in system. For each node data sets its correlation matrix is obtained to see how their three lines are related. Once having the correlation matrix, their

corresponding eigenvalues are computed as shown in section 2.2, such that in this way a signature to each of the different possible component states or fault types of the power system's nodes (*K* in figure 2) are given. The eigenvalues of the correlation matrix for each fault signature, then serve as the training vectors of the PNN corresponding to *Q* as described in section 2.1.

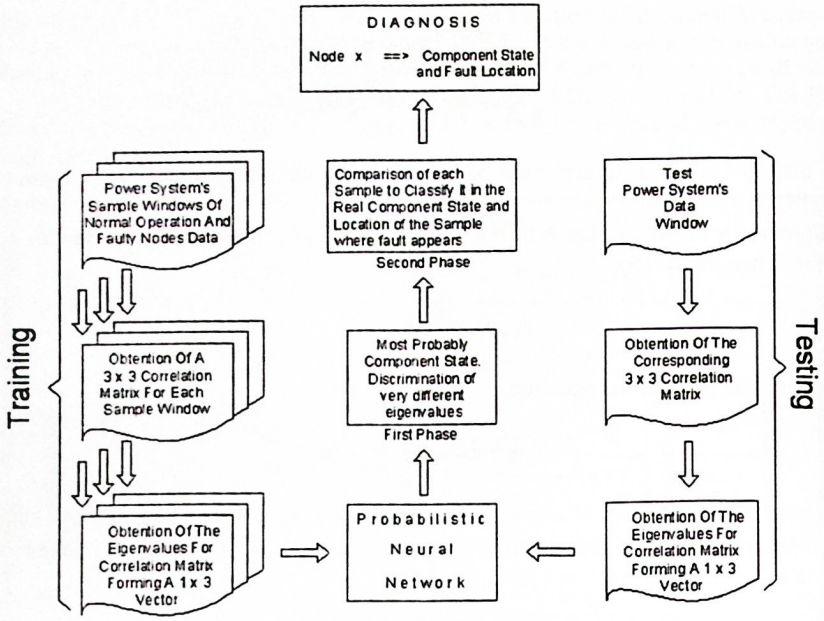


Fig. 3. General fault detection framework.

Each node will have then three eigenvalues (*R* components in figure 2) as they are comming from its correlation matrix that is a *3* x *3* matrix. Then the detection process is carried out in two phases. First phase that serves as a first filter or information discriminator. When there is a system to monitor, a window of *m* samples and *n* power system's nodes is taken, then each node is analyzed separately. *m* samples of the three lines corresponding to a node being monitored to find out which is its most probably state are taken. From the data set corresponding to a node being monitored its correlation matrix and its corresponding eigenvalues are obtained and then these last are used as an input vector to the PNN previously trained. It is mentioned "the most probably state" because unfortunately not all the eigenvalues of all of the node states are so different such that PNN could not classify them easily. But it has been found that for certain signature faults eigenvalues are very similar, thus there is here a

discrimination/classification phase, because it is necessary to look for the real state but only comparing between just a couple of similar signatures instead of the whole bunch of node states. The output from the PNN automatically discriminates node states that are very different and gives the most probably real node state. Once the one possible node state is obtained, a second phase of the framework begins to work. In the second phase each sample of each node is taken and its magnitude is obtained, then a comparison against a constant magnitude of the probably signature faults are carried out. This comparison serves as a classifier that gives the real node component state and can be used too to locate the period of time or sample number where the fault occurs.

4 Case Study

This section shows the performance of the framework proposed in a multiple fault simulations in the IEEE network shown in figure 4.

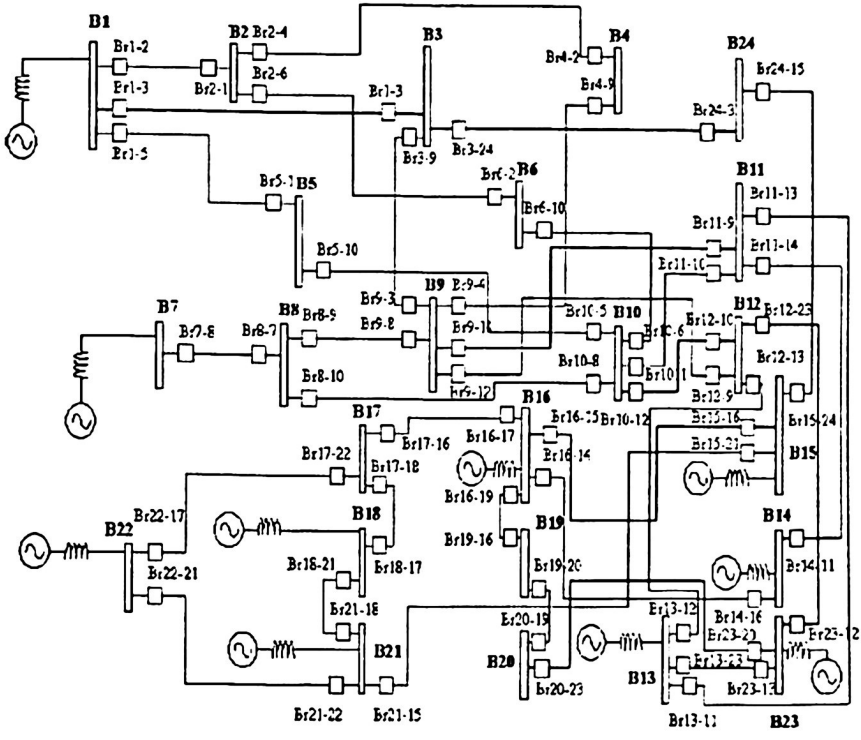


Fig. 4. IEEE reliability test system single line diagram.

Figure 4 shows an electrical power system having dynamic load changes. 24 fault simulations were carried out to determine the performance of the approach including in them symmetrical and unsymetrical faults at random nodes (3,9,10 and 13) taking into account different multiple faults scenarios combining faults such as: one line to ground (A GND), two lines to ground (A-B GND), three lines to ground (A-B-C GND), or faults between two lines (A-B or B-C) and the no fault mode (NO FAULT).

The methodology proposed is applied as follows:

1. Obtain windows of 100 samples from normal and faulty operation history process data (electrical voltage in each node's line).
2. Obtain correlation matrix for each node, which gives a *3* x *3* matrix.
3. Obtain the eigenvalues from the correlation matrix (this gives 3 eigenvalues), with this 3 eigenvalues form an input vector to train the PNN.
4. Take a test data set of 100 samples from the electrical power system being monitored.
5. Obtain correlation matrix for each node, which gives a *3* x *3* matrix.
6. Obtain the eigenvalues from the correlation matrix (this gives 3 eigenvalues), with this 3 eigenvalues form an input vector to the PNN.
7. First Phase: Take the output of the PNN as one of the two probably states of the node monitored.
8. Second Phase: Take each sample of each node monitored and obtain its magnitude, then compare it against the constant magnitude of the two probably signature faults and classify it using this simple criteria. Locate the period of time or sample number where the fault occurs.
9. Give the diagnosis of each node being monitored. If a fault is present in a specific node give the type and location of it, else print NO FAULT.

In the following tables the performance of the approach is shown taking into account three possible cases. Case 1, system is working properly during the first 25 samples from a total of 100, that means 25 samples are ok and 75 samples corresponds to fault present in system. Case 2 takes 50 samples of normal operation data and 50 samples with fault present. And case 3 takes 75 samples of normal operation and 25 with fault present. Table 1 and 2 shows a detailed example of how percentages were obtained. Tables 3 and 4 gives a summary of the obtained percentages for each of the three cases considered.

From tables 3 and 4 it can be said that the different performances are due because eigenvalues of correlation matrices are very similar when there are a major quantity of normal operation data in sample window. The more normal operation data in sample window the more difficult to classify eigenvalues by PNN because they are very similar.

Table 1. Detail performance of detection per node's component state with 25 samples ok and 75 samples with fault present (case 1).

Component State	Correct	Wrong	Accuracy
A-B-C GND	14	0	100%
A-B GND	10	0	100%
A GND	14	0	100%
A-B	18	0	100%
B-C	16	0	100%
NO FAULT	13	11	54.16%

Table 2. Detail performance of detection per node's number with 25 samples ok and 75 samples with fault present (case 1).

Node Number	Correct	Wrong	Accuracy
3	20	4	83.33%
9	19	5	79.16%
10	22	2	91.66%
13	24	0	100%

Table 3. Accuracy of detection per node's component state for the different cases.

Component State	Case 1	Case 2	Case 3
A-B-C GND	100%	100%	100%
A-B GND	100%	100%	100%
A GND	100%	85.71%	92.85%
A-B	100%	83.33%	50%
B-C	100%	68.75%	68.75%
NO FAULT	54.16%	58.33%	79.16%

Table 4. Accuracy of detection per node number for the different cases.

Node Number	Case 1	Case 2	Case 3
3	83.33%	83.33%	83.33%
9	79.16%	75%	70.83%
10	91.66%	87.5%	62.5%
13	100%	95.83%	100%

An important thing to remark is that there has been carried out several tests when all data were from normal operation and it has been found out that the framework have

detected 100% of them as NO FAULT node's component state. Percentages shown in tables are low because criteria used in the second phase of the framework has to be with the maximum magnitude value and a threshold that needs to be set as the upper limit to make the difference between two very similar signatures for the same node.

Figure 5 shows an example of the results obtained for a simulation of case 2, that has the following faults types:

1. 3 A GND, that is a fault present in node 3 of type line A to ground.
2. 9 A-B GND, that is a fault present in node 9 of type line A and B to ground.
3. {10,13} NO FAULT, that is nodes 10 and 13 working properly.

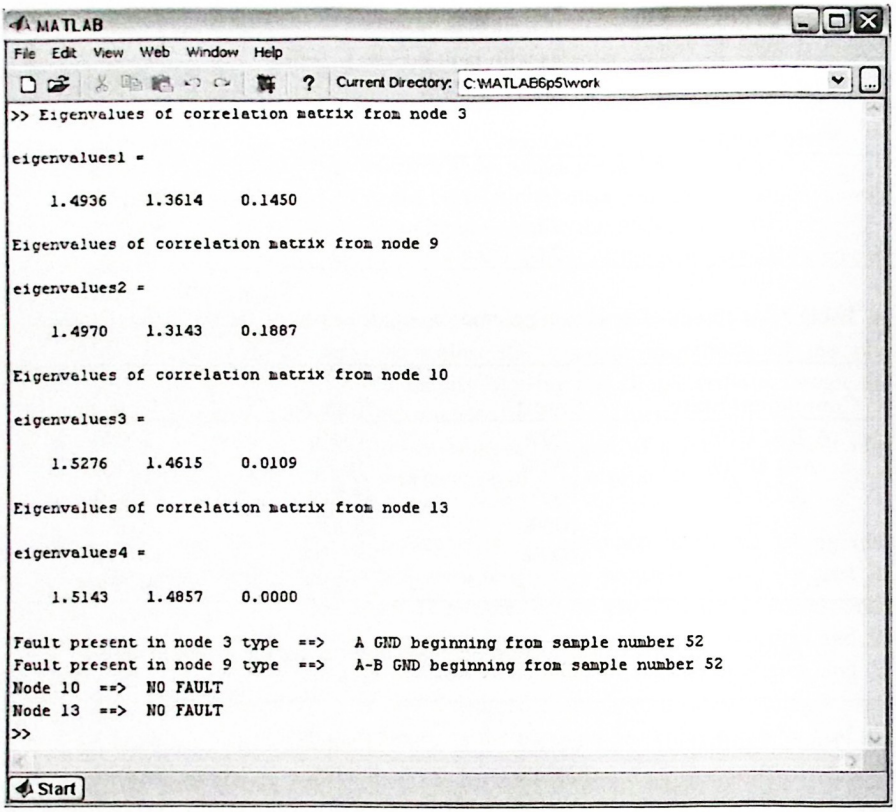


Figure 5. Example of the results given by matlab simulation

4.1 Comparison against a diagnostic system based on probabilistic logic.

To observe the general performance of the proposal, a comparison against a diagnostic system based on probabilistic logic taken from [12] has been carried out. Table 5 and 6 show the performance of the diagnostic system based on probabilistic logic.

Table 5. Performance of detection per node's component of the diagnostic system based on probabilistic logic.

Component State	Correct	Wrong	Accuracy
A-B-C GND	14	0	100%
A-B GND	10	0	100%
A GND	12	2	85.7%
A-B	15	3	83.3%
B-C	16	0	100%
NO FAULT	17	7	70.8%

Comparing the results of both frameworks it can be note that in general they have a very similar performance, but when comparing case 1 of the proposal against the diagnostic system based on probabilistic logic it could be said that framework propose has a better performance. Another important point is that the proposal is relatively easier to implement and to update when power system grows up.

Table 6. Performance of detection per node number of the diagnostic system based on probabilistic logic

Node Number	Correct	Wrong	Accuracy
3	19	5	79.1%
9	21	3	87.5%
10	21	3	87.5%
13	23	1	95.8%

5 Conclusions

This paper has presented a fault detection framework for electrical power systems with dynamic load changes using a PNN based on history process data. An advantage over model based methods is that this framework needs historical data of normal system operation as well as faulty data sets to train the PNN, which in practice it is relatively easy to obtain for computer controlled systems. It has been decided to use the PNN because is ideal when working on classification problems. Its most important

advantage is that it needs only a little time for its training. It has been proposed a two phases relatively easy to implement fault diagnosis method.

In the first phase the eigenvalues of the correlation matrix are taken and used them as inputs for a PNN to classify the node's component state. It has been shown how this classification could be improved and carry out when eigenvalues are very similar with the implementation of a second phase where a simple comparison of each sample magnitude to the constant value of a certain signature fault has been apply and at the same time it gives the location of a fault if a it is present.

It can be concluded too, that as there is as many faulty data in the sample window the proposal has a better performance because eigenvalues are easily classified by the PNN as they have very different values. Another advantage of this proposal is that as it diagnostics each node, it could be detected simultaneous and nonsimultaneous simple, multiples, a combination of different faults as well as their corresponding location on each node separately.

References

1. V. Venkatasubramanian, R. Rengaswamy, K. Yin, S. Kavuri (2003): A review of process fault detection and diagnosis Part I, Part II and Part III. Computers and Chemical Engineering 27(2003) 293-311.
2. Zhang D., Dai S., Zheng Y., Zhang R., Mu P. (2000): Researches and Applications of a Hybrid Fault Diagnosis System Proceedings of the 3r world Congress on Intelligent Control and Automation, Hefei, P.R. China, pp. 215- 219.
3. Z. Yongli, H. Limin, L. Jinling (2006): Bayesian Networks-Based Approach for Power Systems fault Diagnosis IEEE Transactions on power Delivery, Vol. 21, No. 2, pp. 634-639.
4. L. Xu, M. Chow (2005): Power Distribution Systems Fault Case Identification Using Logistic Regression and Artificial Neural Network Proceedings of the 13th International Conference on Intelligent Systems Application to Power Systems, pp. 163-168.
5. H. Ren, Z. Mi (2006): Power System Fault Diagnosis Modeling Techniques based on Encoded Petri Nets IEEE Power Engineering Society General Meeting.
6. M. Peng, W. Hanli, C. Bin (2006): Study of Fault Diagnosis for Power Network based on MAS International Conference on Power System Technology.
7. R. Duda, P. Hart, and D. Stork (2001): Pattern Classification. Second Edition 2001 ISBN 0-471-05669-3.
8. R. Johnson, D. Wichern: Applied Multivariate Statistical Analysis. Prentice Hall, fifth edition.
9. N. Zanzouri, and M. Tagina (2002): A Comparative Study of Hybrid System Monitoring Based on Bond Graph and Petri Net Modeling. IEEE SMC.
10. D. Du, X. Lou, and C. Wu (2005): Dynamic Model of FCCU and its Application in a Hybrid Fault Diagnosis System. 2005 International Conference on Control and Automation (ICCA 2005), June 27-29, 2005, Budapest, Hungary.
11. W. Wang, X. Bai, W. Zhao, J. Ding and Z. Fang (2005): Hybrid Power System Model and the Method for Fault Diagnosis. 2005 IEEE/PES Transmission and Distribution Conference & Exhibition: Asia and Pacific Dalian, China.
12. L. Garza (2001) : Hybrid Systems Fault Diagnosis with a Probabilistic Logic Reasoning Framework. PhD thesis.

Improving Under Sampling with Neural Networks for Class Imbalance Problem

Man-Sun Kim[1]

[1] Research Center for Ubiquitous Information Appliances
Chon-nam National University,
Department of Computer Science, Chon-nam National University
300 Yongbong-Dong, Puk-Ku, Gwangju, 500-757, KOREA
mansun@kongju.ac.kr

(Paper received on June 20, 2007, accepted on September 1, 2007)

Abstract. Data in real world tasks are usually imbalanced, i.e. some classes have much more instances than others. It is one of the reasons that cause the decrease of generalization ability of machine learning algorithms. Therefore, in this paper, we handle the class imbalance problem and proposes an under-sampling method based on SOM (Self Organizing Map), one of neural networks. We apply our methods to UCI Repository data sets that have a class imbalance problem. Finally we show the improvements of new sampling method compared with other sampling methods.

1 Introduction

In a pattern recognition problem, if the number of data belonging to a class is extremely larger or smaller than that of data belonging to another class, there happens the imbalance of data. Class imbalance is often observed in response modeling, remote sensing, image classification, etc. For example, in data used in response modeling, the number of customers who have responded to marketing and purchased goods is much smaller than the total number of customers. What is more, frequently most of important information is included in purchase customers, namely, in the minor class [1,2,3,4,5,6].

In a dataset containing the class imbalance problem, data belonging to the major class are distributed excessively compared to data belonging to the minor class. In such a case, the major class infiltrates into the area of the minor class and has a negative effect on the performance of classification algorithms. Thus, it is essential to solve the class imbalance problem in order to enhance classification performance.

As solutions for the imbalance of data, two types of methods have been proposed as follows [7]. One is using a learning algorithm revised by adding a part that reflects the imbalance of data. This type of methodologies include the method of imposing different penalties on patterns misclassified into the minor class and the major class and the method of adjusting the boundary surface of separation. The other resolves data imbalance by restructuring learning data through sampling. This type of methodologies include largely three methods: over-sampling, under-sampling and ensemble[8]. Sam-

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 151-159

pling-based methods are advantageous over revised algorithm-based methods. Sampling-based methods are easily expandable because they do not revise the algorithm itself but deal with how to compose data used in learning. The reason is that, while revised algorithm-based methods depend on one algorithm, sampling-based methods, if their high performance is proved through a specific algorithm, can be applied immediately to other pattern recognition algorithms.

Sampling is one of techniques for adjusting the size of a training dataset[9,10,11,12,13]. In general, under□sampling and over□sampling are used. Over□sampling replicates minor category data, so it does not add new information through the sampling. In addition, it is efficient in term of time complexity when handling a large volume of data. Under□sampling uses only a part of major category data, so the sample may not represent the characteristics of the whole major category. The ensemble method divides the dataset of the major class into k subsets and forms a learning dataset by combining each subset with the data of the minor class. The classifier is trained with each of the k learning dataset and the data are combined in several gathering methods of ensemble. This method shows improvement in performance experimentally, but it has disadvantages such as low accuracy of used problems, the possibility of distorted distribution of data due to the use of simple separation, and difficulty in maximizing the advantage of ensemble because the population of ensemble depends on the imbalance ratio. Under-sampling uses only a part of the major class data and thus it cannot represent the entire data of the major class. Particularly when the data difference between the minor class and the major class is large, data distribution is distorted severely. Thus, although under-sampling enhances performance, the results of individual experiments vary significantly. Many of recent under-sampling methods under study select the major class using specific strategies [8,17].

In order to solve the problems in under-sampling, the present study proposes a method of selecting major class data useful in learning using the mechanism of biological competition. The method clusters data of similar characteristic using a clustering method rather than random sampling of major class data and then builds learning data limitedly. The proposed method obtains meta-data reexpressed in competition relation by SOM (self-organizing map), which is a method of unsupervised learning, and deletes major class data around the minor class. This method is advantageous in that it can reflect the characteristics of the entire data using data distribution, and decreases data size through sampling.

This paper is composed as follows. Chapter 2 reviews recent research trends, and Chapter 3 discusses a novel under□sampling techniques based on neural network. Chapter 4 conducts an experiment and analyzes the results, and Chapter 5 draw conclusions.

2 Related works

Kubat and Matwin[14] also selectively under-sampled the majority class while keeping the original population of the minority class with satisfied results. The majority examples were divided into four categories: some noise overlapping the positive class decision region, borderline samples, redundant samples and safe samples. The result of

Kubat's experiment showed considerable improvement in performance, but not if the degree of imbalance was high.

Batista et al[15] used a more sophisticated under-sampling technique in order to minimize the amount of potentially useful data. The majority class instances are classified as "safe", "borderline" and "noise" instances. Borderline and noisy cases are detected using Tomek links, and are removed from the data set. Only safe majority class instances and all minority class instances are used for training the learning system.

Japkowicz [16] discussed the effect of imbalance in a dataset. She mainly evaluated two strategies: under-sampling and resampling. Two re sampling methods were considered. Random resampling consisted of resampling the smaller class at random until it consisted of as many samples as the majority class and "focused resampling" consisted of resampling only those minority instances that occurred on the boundary between the minority and majority classes. Random under-sampling was also considered, which involved under-sampling the majority class samples at random until their numbers matched the number of minority class samples; focused under-sampling involved under-sampling the majority class samples lying further away. She noted that both the sampling approaches were effective, and she also observed that using the sophisticated sampling techniques did not give any clear advantage in the domain considered.

Kim et al[17] proposed a focused sampling method which is more superior than previous methods. To solve the problem, The proposed method must select some useful data set from all training sets. To get useful data set, The proposed method devide the region according to scores which are computed based on the distribution of SOM over the input data. The scores are sorted in ascending order. They represent the distribution of the input data, which may in turn represent the characteristics of the whole data. A new training dataset is obtained by eliminating unuseful data which are located in the region between an upper bound and a lower bound. The proposed method gives a better or at least similar performance compare to classification accuracy of previous approaches.

3 The proposed methods

Because under-sampling uses only a part of major category data, the sample cannot represent the whole major category and distorts data distribution severely. To solve these problems, this study proposes an under-sampling method based on SOM (Self Organizing Map), one of neural networks.

For under-sampling, the present study used SOM, a method of unsupervised learning. As a result, we can reexpress high-dimensional data distribution two-dimensionally. Each data reexpressed into meta-data is allocated to a grid on the two-dimensional map. Because a grid can contain one or multiple data, it is very efficient to express data compactly. In order to minimize heterogeneity among grids containing information on different classes, grids adjacent to the minor class are selected and the selected major class grids are removed. Because the selected major class grids contain a large number of data, this process can reduce the number of data samples considerably.

3.1 SOM learning selecting the best-matching unit (BMU)

In Kohonen's learning, each neuron calculates the connection strength vector and the distance to the input vector. In addition, each neuron competes with others for the privilege of learning, and the closest neuron (best-matching unit) wins the competition. This is the application of the biological competition mechanism to learning. The best-matching unit is the only neuron that can send output signal. In Kohonen's learning, only the winner can issue output, and the winner and its adjacencies can adjust their connection strength. Because clustering is not the objective of this study, we do not need learning for adjusting connection strength. That is, only the method of selecting the best-matching unit is applied to under-sampling.

When the construction of a Kohonen network require three tasks, which are generally not necessary in other types of neural networks. One is initializing the connection strength vector of neurons on the layers adequately with random values. In the present study, it was initialized with [1,1,1,.....,1] according to the number of attributes. Another task is using normalized values between 0 and 1 for the connection strength vector and input vector. The other task is determining the size of the 2-dimensional grid. In general, the size of the grid is determined based on [ceil(5*$\sqrt{datasamples}$)] and is a multiple of the number of attributes, and with the size of the grid, the number of rows of the grid is calculated. For example, if the number of data is 200 and the number of attributes is 8, grid size becomes 72 [9*8]. The three factors are very important values to be emphasized in Kohonen network, but this study used values obtained through trials and errors. This part shall be discussed later.

The process of selecting the best-matching unit is as follows. In the process, we can obtain the result that the connection strength vector of the best-matching unit is closest to the input vector.

(1) Present a new input vector.
(2) Calculate the distance between the input vector and each neuron.
(3) Find the position of the closest grid among the calculated distances.

In order to select the best-matching unit, the distance between input and output neurons is calculated. The distance is calculated using Euclidean distance. To find the best-matching unit, the unit and distance of all maps are calculated for each data vector.

```
for i=1:dlen,
  for j=1:munits,
   for k=1:dim,
     Dist(j,i)=Dist(j,i)+mask(k)*(D(i,k)-M(j,k))^2;
   end
  end
end
(dlen:data samples, munits:row of grid, dim:column of
grid Dist(j,i): distance from i to j, mask(k): mask is
the weighting vector for distance calculation, M:codebook
matrix, D: data matrix)
```

Fig. 1. Calculation of distance between the input vector and each neuron.

3.2 Elimination of adjacencies (from the units of the minor class)

The proposed method is as follows. After the position of the closest grid among the calculated distances is obtained as a result of 3.1, each unit of the minor class is named uniti. Then, adjacent units uniti-1 and uniti+1 are stored in the eliminated unit candidate set. Here, in order to store more units in the candidate set according to the characteristic of data, the condition can be changed to uniti-2, uniti+2, ...uniti-c, uniti+c.

```
Uniti : results of 3.1
C : candidate sets of elimination units
n : number of units
for i= to n
if (i=unit of the minor class)
C=C+adjacency unit(uniti-1, uniti+1)
end if
return C
```

Fig. 2. Generation of the eliminated unit candidate set.

The generated candidate units to be eliminated are removed from the units and, as a consequence, data units of the major class, which are necessary for learning, are obtained.

The figure below shows the three steps for obtaining the units of the major class. The step of initial map generation and the step of allocation to each grid were explained in 3.1. For example, let's assume 100 input data and map size of [7 4]. Each data is allocated to each grid and forms a sub cluster. This is called a unit. As a result, the number and ratio of major class data can be reduced by eliminating units adjacent to the minor class (in Figure 3, red units). In this example, the minor class has 13 data and the major class has 87 data in the step of grid allocation. In the next step, the number of major class data can be reduced from 87 to 56 by eliminating units adjacent to the minor class. Even more adjacent units can be eliminated by the researcher's judgment. This criterion should be considered together with the characteristic of data. It is because the form of adjacent data may be different from the figure below depending on the grid distribution of the minor class and the major class. If the units of the minor class are distributed unevenly or major class data and minor class data share the same unit, units to be eliminated for high performance should be decided carefully.

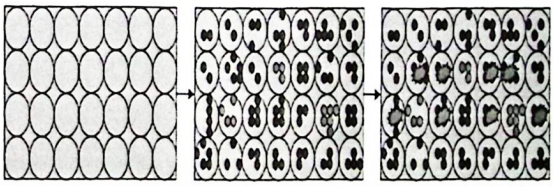


Fig. 3. Map formation step-nit allocation step-eliminate step.

In the figure above, the map formation step and the unit allocation step were explained in 3.1. In 3.2, major class units adjacent to the minor class are eliminated.

The method proposed in 3.2 is different from the method proposed by Kim et al. [17]. Their method generated data as in 3.1 but divided the result of reexpressed distribution into two sections (upper and lower). Data between the two sections were regarded as unuseful patterns and were not used in learning. This method can decrease the imbalance ratio to some degree but it may eliminate useful patterns. That is, if minor class data are not included in the two sections, the sampling method is meaningless.

4 Experiments

Because under-sampling uses only a part of major category data, the sample cannot represent the whole major category and distorts data distribution severely. To solve these problems, this study proposes an under-sampling method based on SOM (Self Organizing Map), one of neural networks. In this section, we evaluate the performance of the proposed approach in the various domain. the experiments were performed with Matlab 7.0, on 1GB RAM, Pentium IV processor. For experiment on the proposed method, we used the data of the UCI Repository [18] available on the Web.

Table 1. Data sets used in the experiments.

data set	examples	attributes	class(min.maj.)	class(min.maj.)(%)
pima	768	8	(1,0)	35.02%, 64.97%
breast	683	10	(1,0)	34.99%, 65.00%
glass	214	9	(ve_win_float_proc, remainder)	7.9%, 92.5%
wine	178	13	(3, remainder)	26.96%, 73.03%

In this experiment, we used the 5-fold hold-out method and the cross-validation method for the accurate measurement of the result of the experiment on the proposed method. The experiment divides experiment data into 5 partial datasets. One of them is used in validation, and the other 4 are used in learning. In this study, kNN was used as a basic classifier to evaluate performance. The performance scale should reflect data imbalance. In most of pattern recognition algorithms, simple accuracy measured as follows is used as a performance scale.

$$\text{Accuracy}=\frac{TP+FN}{TP+TN+FP+FN}$$

In case there happens data imbalance, however, simple accuracy cannot reflect the accuracy of the minor class sufficiently. For example, let's assume that only 10 (1%) out of 1000 customers purchased and the other 990 (99%) did not purchase. If the response model judges that none of the customer purchased, the simple accuracy of the response model is 99%, which is seemingly very high performance. However, the model failed to distinguish purchase customers, who are important, and thus it is meaningless. In this study, we use G-Mean as a performance scale for the balanced consid-

eration of the accuracy of the minor class and the major class. G-Mean (Geometric Mean) is a scale that can consider the minor class and the major class equally. It is measured as follows, and with this, the accuracy of the two classes can be understood as a geometric mean.

$$\text{G-Mean}=\sqrt{\frac{TP}{TP+FN}\times\frac{TN}{FP+TN}}$$

Table 2. The accuracy (Accu, G-Mean).

		pima	breast	glass	wine
No sampling	Accu	0.7292	0.9484	0.9679	0.9244
	G-Mean	0.6731	0.9602	-	-
kim [17]	Accu	0.4260	0.9326	0.6759	0.8202
	G-Mean	0.6457	0.9216	0.2843	0.3333
proposed	Accu	0.9055	0.9025	0.8539	0.9244
	G-Mean	0.8664	0.9860	0.6667	0.3333

As a whole, the proposed method showed high G-mean, enhancing the accuracy of both the two classes. The method proposed by Kim et al. [17] divided the result of reexpressed distribution into two sections (upper and lower), each of which contained 25% of the data. Here, discarded data may contain important information. In order to minimize the loss of important data, we eliminated data on the boundary between the minor class and the major class rather than selecting data to be eliminated by fixing the sections of the classes. Major class data on the boundary were removed by the researcher's judgement. Clarifying this part is one of tasks to be studied in the future.

In case of glass and wine, performance was very low. G-mean could not be calculated because minor class data could not be classified properly when classification was made without sampling. The reason for this phenomenon is believed to be that, in the process of imbalance data generation, one class was used as the minor class and the other classes were used as the major class because the data contained three or more classes. This shows that the result of experiment depends on the characteristic of data.

Table 3. The result of calculating imbalance ratios.

	samples	pima(768)	breast(683)	glass(214)	wine(178)
No sampling	Min	268	239	17	48
	Maj	500(65.10%)	444(65.00%)	197(92.06%)	130(73.03%)
kim [17]	Min	135	132	9	18
	Maj	250(64.93%)	209(61.29%)	99(91.66%)	71(79.77%)
proposed	Min	126	229	17	48
	maj	350(73.52%)	228(49.89%)	79(82.29%)	71(59.66%)

The table above shows the result of calculating imbalance ratios. The ratio of major class data decreased in general except PIMA. The imbalance ratio of pima data did not

have a significant effect on the accuracy of the major class and the minor class. Rather, as shown by G-mean, the overall accuracy and the accuracy of the minor class were improved.

The present used SOM to reexpress the center of class distribution. This method learns input data given without any specific instruction and expresses the characteristic of the data using specific output neurons in the space. Accordingly, we can have advantages as follows by using the results in selecting under-sampling learning data for solving the class imbalance problem.

First, the ratio of class imbalance data can be decreased. Because the distribution of the entire data is reexpressed in summarized units and sampling is made from data within a specific area, the imbalance ratio can be decreased considerably.

Second, this method can solve the problem that sampled data do not represent the whole data. In random under-sampling, the distortion of data distribution is worsened with the increase of size difference between the major class and the minor class. However, the representativeness of data can be preserved through under-sampling by a specific strategy.

Third, this method is easily expandable for learning algorithms. Because it provides the method of composing data to be used in learning, it is applicable to any learning algorithm with improved performance.

5 Conclusion

The present study proposed an efficient data selection method for solving the problem of class imbalance. In this study, first, the distances between input and output neurons was calculated to select the best-matching unit by applying SOM, which is unsupervised learning. This step finds the position of the closest grid among the calculated distances. Second, by eliminating units adjacent to minor class data, we obtained more homogeneous major class data and more heterogeneous redistribution of major and minor class data. That is, the problem of class imbalance was solved by using only useful data. The improved concentrated sampling method has advantages as follows. First, the class imbalance ratio is reduced. Second, the problem that sample data cannot represent the whole data is solved. Third, expandability for learning algorithms is high.

References

1. S. Cho, H. Shin, E. Yu, K. Ha, and D. MacLachlan, "Data Mining Problems and Solutions for Response Modeling in CRM," Entrue Journal of Information Technology, Vol.5, No.1, (2006) 55-64.
2. L. Bruzzone, D. Fernàndez Prieto, "A Combined Supervised and Unsupervised Approach to Classification of Multi Temporal Remote Sensing Images," IEEE 2000 International Geoscience and Remote Sensing Symposium (IGARSS), Honolulu, Hawaii, 24-28, Vol. 1, (2000) 162- 164.
3. R. Yan, Y. Liu, R. Jin, A. Hauptmann, "On Predicting Rare Classes With SVM Ensembles In Scene Classification," IEEE International Conference on Acoustics, Speech and Signal

Processing (ICASSP), (2003) 21-24.
4. Guobin Ou and Yi Lu Murphey, "Multi-class pattern classification using neural networks," Journal of Pattern Recognition, Vol 40, Issue 1, (2007) 4-18.
5. Vicenc Soler, Jesus Cerquides, Josep Sabria, Jordi Roig, Marta Prim, Imbalanced Datasets Classification by Fuzzy Rule Extraction and Genetic Algorithms, IEEE International Conference on Data Mining - Workshops (ICDMW'06), (2006) 330-336.
6. Yang Liu, Nitesh V. Chawla, Mary P. Harper, Elizabeth Shriberg and Andreas Stolcke, "A study in machine learning from imbalanced data for sentence boundary detection in speech," Journal of Computer Speech & Language, Vol 20, Issue 4, (2006) 468-494.
7. Yanmin Suna, Mohamed S. Kamela, Andrew K.C.Wongb,YangWangc, "Cost-sensitive boosting for classi?cation of imbalanced data," Journal of Pattern Recognition.2007
8. Pilsung Kang and Sungzoon Cho, "EUS SVMs: Ensemble of Under-Sampled SVMs for Data Imbalance Problems", Proceedings of International Conference on Neural Information Processing 2006
9. Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W. , SMOTE : Synthetic Minority Oversampling Technique, Journal of Artificial Intelligence Research 16, (2002)321-357.
10. Japkowicz, N. , "The Class Imbalance Problem : Significance and strategies," International Conference on Artificial Intelligence, 2000.
11. Chawla, N.V., Hall, L. and Kegelmeyer, W. , "SMOTE : Synthetic Minority Oversampling Techniques," Journal of Artificial Intelligence Research 16, pp321-357, 2000.
12. Maloof M. A. , "Learning when Data Sets are Imbalanced and When Costs are Unequal and Unknown," ICML Workshop on Learning from Imbalanced Data Sets II, 2003.
13. Chawla, N. V. , "C4.5 and Imbalanced Data sets: Investigating the effect of sampling method, probabilistic estimate, and decision tree structure," ICML-2003.
14. Kotsiantis S., Pierrakeas C., Pintelas P., "Preventingstudent dropout in distance learning systems using machine learning techniques," AI Techniques In Web-Based Educational-Systems at Seventh International Conference on Knowledge-Based Intelligent Information & Engineering Systems, (2003) 3-5.
15. Batista G., Carvalho A., Monard M. C. , "Applying One-sided Selection to Unbalanced Datasets," In O. Cairo, L. E.Sucar, and F. J. Cantu, editors, Mexican International Conference on Artificial Intelligence (MICAI), (2000) 315–325.
16. Japkowicz N., "The class imbalance problem:Significance and strategies," International Conference on Artificial Intelligence, Las Vegas, 2000.
17. Man-sun Kim, Hyung-Jeong Yang, Soo-Hyung Kim, Wooiping Cheah, "Improved Focused Sampling for Class Imbalance Problem," Journal of Korea Information Processing Society, No.14(b), Vol.4 (2007).
18. http://www.ics.uci.edu/~mlearn/databases/
19. Jigang Xie, Zhengding Qiu, "The effect of imbalanced data sets on LDA:A theretical and empirical analysis," Journal of Pattern Recognition, (2007)557-562.

Closing Price Prediction for Auctions of Hotel Rooms on the TAC Classic

Eber Jair Flores Andonegui[1], Darnes Vilariño Ayala[1], Fabiola López y López[2]

[1] Facultad de Ciencias de la Computación, BUAP
Puebla, México
[2] Dirección de Modalidades Alternativas de Educación, BUAP
Puebla, México
eberjair@hotmail.com darnes@cs.buap.mx, fabiola.lopez@siu.buap.mx

(Paper received on June 20, 2007, accepted on September 1, 2007)

Abstract. In this paper a brief description of the Trading Agent Competition (TAC) Classic and the problems that agents must face are presented. Moreover, episodes are proposed as a solution to predict the closing price of hotel rooms for a bid in the competition. To do that, a five layer neural network is used and, after upgrading the original neural network twice, the results, obtained from 20 games among dummy and other agents are shown.

1 Introduction

The Trading Agent Competition (TAC) has been organized during many years ago. In this competition, travel agents compete against each other in order to better satisfy the requirements of their customers. What makes the game interesting is the fact that every trading agent is a software agent and, it is specialized on selling travel packages to determined customers. Each travel package includes flights and hotel room reservations as well as tickets for entertainment activities of the customers [1].

The main problem is that every travel agent must propose to its customer the best day to travel, according to the conditions of the market, the customer requirements on hotel quality and desired entertainment and, flights availability. Travel agents work in an autonomous way, i.e. they must be able to take decisions on their own. To do so, an architecture that includes the following components has been proposed: (a) learning schemas that allow agents to adapt their behavior to the current environment, (b) mechanisms to interact with the platform and other participants and (c) capabilities to analyze the changing conditions on the market.

Many papers have been written about the TAC Classic, and some of the production of 2004 and 2005 is referred. For example, Sardinha et al 1 present a multiagent architecture that solves this problem. Here, each agent of the system specializes in a part of the market, and it can predict the purchase prices of tickets in the following next days by checking the room availability. This multiagent system considers a learning architecture that is supported on a knowledge base which is created during the game time. Pannos Toullis et al2 approach the problem by using fuzzy logia and reasoning rules to predict the room prices, by considering available historical data. Trying to efficiently assign the amusement shows, they design a strategy that guarantees the requirements of

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 161-172

each client. Michael P. Wellman et al. on 3 proposes a hierarchical distribution of the original problem, by analyzing and predicting the behavior of a rival agent. Against one rival, the Walverine agent may predict the whole of its behavior. Against two rivals, the agent may predict a 74.1% of their behavior. But if the rivals are extended to four of them, the agent may predict only the 2.4% of their behavior. In general, those proposals consist on multi-agent systems that allow the specialization of each subagent on a part of the whole market.

Our research has been done in three different phases: an analysis of the conditions of the environment where agents participate; a study of the platform where the competition takes place; and the design and implementation of a Neural Network that can predict the closing bidding prices for the hotel rooms. The paper is structured as follows: Section 2 gives a brief description of the TAC Classic. Section 3 describes the main problems of the game. Section 4 shows the design of the neural network that can predict the closing bidding prices for the hotel rooms. Section 5 presents the results for the local tests over the neural network itself. Finally, Section 6 contains the conclusions about our results.

2 The TAC Classic Description

The TAC Classic provides a platform where simulations of games can be done. The platform is in charge of the following tasks: hosting the participants; activating the agents once the simulation starts and removing them once the simulation ends; and providing relevant information from each round. Some of the data obtained from the platform are used as configuration values to participate on further simulations. These data consist on the bidding for hotel rooms, flights and entertainment shows prices.

In each TAC Classic game, 8 travel agents participate. These agents organize travel tours from TACTown to Tampa for eight different customers during an imaginary period of 5 days. Customers express their preferences for every aspect of the trip, Their preferences are represented by a utility function [1]. The goal of each agent is to maximize the total utility of their customers. Agents compete against each other to obtain the best result on assembling the tours of their clients. They participate on 28 bids. A tour consists of a round flight, hotel reservation and tickets for entertainment events such as the alligator wrestling, an amusement park or a museum.

Each one of these goods –flights, hotel reservations and entertainment events- are traded in separate markets with different rules as following:

There are two hotels in Tampa: Tampa Towers (TT), which has a high price because it is the most comfortable hotel, and Shoreline Shanties (SS) which is the cheapest option for traveling to Tampa. Once the hotel has been assigned, it cannot be changed during the staying of the client in Tampa. Since the client needs hotel on his arrival night and until the pre-departure night, there is no hotel available on the last day of game. To get a room, the agent must participate on ascending auctions: one auction for each combination of hotel and night, each with 16 rooms being auctioned off. The price of each hotel is based on the customer preference for the Tampa Towers Hotel. The agents are not allowed to exchange hotel rooms between themselves. For a deeper

specification of the other parameters such as flight reservations and entertainment events, we recommend to check [1].

Regarding hotel rooms, each agent must send an offer for the auction. While it is open, the auction price is updated with the current values. All of the bids are presented as a pair (q, p) where q is the quantity, and p is the price offered by the agent. If q is less than zero, then it is indicated that the agent wants to 'sell' a quantity q of tickets on a base price p. If q is over zero, then the agent is interested in 'purchasing' certain quantity q of tickets by paying p. The client preferences are specified as follows: The day when they wish to arrive to Tampa (PA); the departing day from Tampa (PD); a prize for assigning the best hotel (HP) -this value is between $50 and $150-; and a prize value for assigning an entertainment type -on a range between $0 and $200 according to the customer preferences: AW for alligator wrestling, AP for amusement park, and MU for museum.

The tour is specified by using these variables: the arriving day (AA), the departure day (AD), and a binary value (0 or 1) that indicates whether the Tampa Towers was selected (TT?).

The tour is feasible if it contains the hotel reservations for each night between the arrival and departure dates. The shortest stay on Tampa is for one day, and obviously the customer will not stay at the hotel while flying back to TACTown. It is important to remark that the obtained prize for reserving the Tampa Towers on a travel is for the whole tour and not for each night spent in Tampa.

The winning agent is the one that obtains the highest score of the game. The score is obtained from the obtained utilities less the costs for each tour. The profit of each client is measured in dollars and is calculated by the following utility function:

$$u = 1000 - travel_penalty + hotel_bonus + fun_bonus \quad (1)$$

$$\text{where: } travel_penalty = 100*(|AA - PA| + |AD - PD|);$$

$$hotel_bonus = TT? * HP; \text{ and}$$

$$fun_bonus = AW?*AW + AP?*AP + MU?*MU.$$

The parameter *hotel_ bonus* can be $0 if the client is not assigned to the Tampa Towers (TT?=0) for staying. If he is assigned (TT?=1) the hotel price will take the HP value defined previously, and it is added to the general utility.

3 Problems to be solved by the Agent

Agents should be able to offer clients optimized tour packages that satisfy most of their needs. Therefore, it is important for agents to get a reservation at the Tampa Towers in such a way that the pair *arrival-departure* days are in accordance with the desired staying days. In addition, agents must also get the required entertainment events for the staying days in Tampa.

One way to solve the problem is by programming separate modules that solve a part of the whole problem (i.e. dividing the problem into sub-problems, see Figure 1). Then, the agent makes its own decisions by considering the solutions found by each module.

Each module has been designed to operate different kinds of transactions, such as the hotel rooms by using English auction and the flight prices by using a stochastic function. Once the agent predicts the next possible scenario, it consults the possible strategies and the tasks that it should perform to confront the stage, and finally it saves the results of applying the current strategy. Once the decisions are made, the offers are sent to the server. One of the most important problems during the game is to guarantee a room for the clients during their stay in Tampa. To solve this problem, an artificial neural network is proposed, such as He et al [5] did.

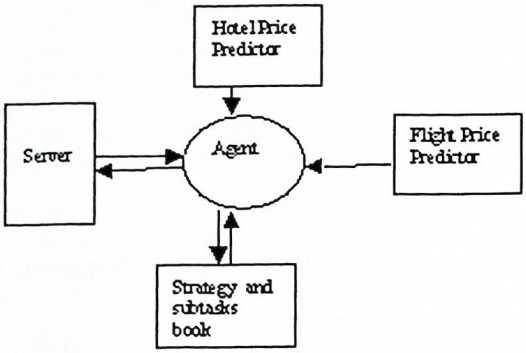


Fig. 1. Problem solution by programming separate modules.

To get the hotel rooms the agent must participate in auctions. The format of the auctions for the hotel rooms is English Standard, it means ascending from one to many. The only modification is that the auctions are closed randomly on every minute of game. Each auction is closed randomly during the eight minutes of the game. The agent does not know previously which auction will be closed during the next minute. Each hotel offers 16 rooms per night, on a minimum price of $0.

The requested price is announced every minute, and it is calculated by considering the sixteenth highest price of all the bids. When an agent makes a new bid, the suggested price must be higher than the one established, by considering the following rules: the offered price of a new bid must be at least one unit higher than the requested price. If the current offer acquires q rooms, the new offer must request at least q rooms, at least one unit higher than the requested price. Agents cannot retreat their bids. When an auction is closed, the 16 units with the highest offered prices will be sold and agents will pay the requested price. Since the requested price is known only every minute of the game, it is not sure that for an offered price, that the agent can obtain the required rooms. A possible solution can be found by trying to predict the requested price and make decisions about how much to spend for the required rooms.

4 Designing the Neural Network

The factors that are considered to predict the closing price of an auction are, first, the customer preference for the hotel type (*Pref*). A higher preference for the room tends to increase the offered price for it. *Pref* is fuzzily defined with the values *Low, Medium* and *High*. We also consider the average closing price on previous auctions for the same room (*Prom*). Some rooms tend to be more required, so the offering price could be increased. The fuzzy values for this variable are *Low, Medium* and *High*. Another factor that is considered is the distribution of the closed auctions (*Dist*). If the auctions of the hotel rooms on the closer days to the current auction are closed, the price for the same room may tend to increase because clients cannot change the hotel during their staying at Tampa. The fuzzy values considered are *Spread, Half* and *Gathered*. The only values this variable can take are (a) 0 if no auction from the two nearest auctions to the current day has closed; (b) 1 for one of the two auctions that has closed; and (c) 2 if both auctions are closed. For the expected price (*Price*) five fuzzy values were considered: *Very_Low, Low, Medium, High* and *Very_High*. We create 27 rules of fuzzy reasoning and a 5-layer neural network.

The values for the input variables correspond to the first layer. Each node in this layer returns a belonging value for the input. For the cases of the values of Pref and Prom, a normal distribution functions is given as follows:

$$S_i^{(1)} = e^{-\frac{(x-\bar{x}_i)^2}{2\delta_i^2}} \quad \text{for } i = 1\ldots6 \tag{2}$$

Where x is the input value and the parameters $\bar{x}$ and δ are fixed when the neural network learns from its environment in order to reduce the prediction error for all of the corresponding nodes to the variable *Pref*. For all of the nodes related to the variable *Prom*, the parameters $\bar{x}$ and δ are calculated as follows:
Let MB to be the lowest average price, and CS the highest average price, then:

$$\bar{x}_i = \begin{cases} \dfrac{CS+5MB}{6} & i=4 \\ \dfrac{CS+MB}{2} & i=5 \\ \dfrac{5CS+MB}{6} & i=6 \end{cases} \tag{3}$$

$$\delta_i = \begin{cases} \bar{x}_i - MB & i=4 \\ \bar{x}_i - \dfrac{CS+2MB}{3} & i=5 \\ CS-\bar{x}_i & i=6 \end{cases} \tag{4}$$

It can be observed that the assignment of values for these parameters depends on the general behavior of the auctions, and it does not depend on the neural network itself. It means that the assignment is not fixed through the learning process. To determine the factor *Dist*, a discrete activation function is used, it is given by:

$$S_i^{(1)} = \begin{cases} a_i & \text{if } x = 0 \\ b_i & \text{if } x = 1 \\ c_i & \text{if } x = 2 \end{cases} \quad \text{for } i = 7...9 \tag{5}$$

Where the values a, b and $c \in (0,1]$, and they are fixed when the neural network learns to decrease the prediction error. The second layer belongs to the inference rules. This layer determines the strength of the rule by multiplying all of the inputs:

$$S_i^{(2)} = p_i = \prod_{j \in C_i^{(1)}} S_j^{(1)} \text{ for } i = 1...27 \tag{6}$$

Where $C_i^{(1)}$ is the group of nodes from the first layer, that pass information to the i node on the second layer. The third layer calculates the importance degree for each rule, or in other words, the relative strength of each rule by following the function:

$$S_i^{(3)} = p'_i = \frac{p_i}{\sum_{j \in C^{(2)}} p_j} \tag{7}$$

Where $C^{(2)}$ is the group of nodes from the second layer that correspond to the i node. The fourth layer accumulates the weights of each rule, by considering the same output, to generate the value for the output variable, by applying the following function:

$$S_i^{(4)} = r_i \sum_{j \in C_i^{(3)}} p'_j \tag{8}$$

Where $C_i^{(3)}$ is the group of nodes from the previous layer that corresponds to the i node of the current layer. Finally the fifth layer gathers all the values of the fuzzy output variables from the fourth layer, with their assigned weight as follows:

$$S^{(5)} = \sum_{i \in C^{(4)}} (r_i \sum_{j \in C_i^{(3)}} p'_j) \tag{9}$$

Where $C^{(4)}$ is the group of nodes from the fourth layer. Once the price is obtained on the output of the neural network, if it is smaller or equal to the requested price in the current auction, then the requested price is taken, and increased over a 10%, as the next offering price. A second version of the same neural network can be obtained by considering the current asked price *Asked_Price* as an input on the first layer, instead of the customer preference *Pref*. Using the same fuzzy rules the functioning of the new network is explained as follows: A current asked price *Asked_Price* means that some agents are pursuing the same auction, and then it tends to increase the next requesting price. During the training of the neural network the parameters are adjusted, in order to guarantee the learning of the network itself. The main objective is to minimize the error value, which is calculated by:

$$E = \sum_j \frac{1}{2} (Y_j - S_j^{(5)})^2 \tag{10}$$

Every time that a new group of data is received, the output values of all the nodes are saved, in order to calculate $\frac{\partial E}{\partial S}$ of each node. These values are considered on the application of the optimizing algorithms to minimize the errors on the calculations. The first adjusted parameters are the r_i parameters that are on the fourth layer. The applied technique, as applied on the paper of He et al. [5] is based on the optimal direction of maximal decrease:

$$r(t+1) = r(t) + \eta(-\frac{\partial E}{\partial r}) \tag{11}$$

Where η is the learning factor, that is initially considered as $\eta = .01$. The inference rules to adjust the parameters r_i in the fourth layer is:

$$\frac{\partial E}{\partial r_i} = \frac{\partial E}{\partial S^{(5)}} \frac{\partial S^{(5)}}{\partial S_i^{(4)}} \frac{\partial S_i^{(4)}}{\partial r_i} = (S^{(5)} - Y) \sum_{j \in C_i^{(3)}} p'_j \tag{12}$$

Then, r_i is updated as follows:

$$r_i(t+1) = r_i(t) - \eta(S^{(5)} - Y) \sum_{j \in C_i^{(3)}} p'_j \tag{13}$$

The following parameters that must be adjusted are $\bar{x}$ and δ for all the nodes of the first layer that correspond to the input variable *Pref*. The implemented technique is based on the optimization by targeted gradients. Let it be g_t the gradient of the iteration t (t>1), then the new search direction is obtained by combining the current maximum descent direction against the previous direction, that is:

$$p_t = -g_t + \beta_t p_{t-1} \tag{14}$$

By using Fletcher-Reeves's Upgrade it is obtained:

$$\beta_t = \frac{g_t^T g_t}{g_{t-1}^T g_{t-1}} \tag{15}$$

Then for modifying the learning rules for the parameter $\bar{x}$, a gradient that considers the next partial derivative as the error function –where $C_{-i}^{(2)}$ is the group of nodes from the second layer that are connected with the i node of the first layer as follows:

$$\frac{\partial E}{\partial \bar{x}_i} = \sum_{m \in C_{-i}^{(2)}} \left(\frac{\partial E}{\partial S_m^{(2)}} \frac{\partial S_m^{(2)}}{\partial S_i^{(1)}} \frac{\partial S_i^{(1)}}{\partial \bar{x}_i} \right) \tag{16}$$

$$\frac{\partial E}{\partial \bar{x}_i} = \sum_{m \in C_{-i}^{(2)}} \left(\sum_{k \in S_{-m}^{(3)}} \left(\frac{\partial E}{\partial S_k^{(3)}} \frac{\partial S_k^{(3)}}{\partial S_m^{(2)}} \right) \frac{\partial S_m^{(2)}}{\partial S_i^{(1)}} \frac{\partial S_i^{(1)}}{\partial \bar{x}_i} \right) \tag{17}$$

where:

$$\frac{\partial E}{\partial S_k^{(3)}} = \left(S^{(5)} - Y\right)_k; \quad \frac{\partial S_k^{(3)}}{\partial S_m^{(2)}} = \begin{cases} \dfrac{\sum_k p_k - p_m}{\left(\sum_k p_k\right)^2} & do\ if\ k = m, \\ \dfrac{-p_m}{\left(\sum_k p_k\right)^2} & do\ if\ k \neq m, \end{cases}$$

$$\frac{\partial S_m^{(2)}}{\partial S_i^{(1)}} = \frac{p_m}{S_i^{(1)}}; \text{ and } \frac{\partial S_i^{(1)}}{\partial \bar{x}_i} = e^{-\frac{(x_i - \bar{x}_i)^2}{2\delta_i^2}} \frac{\left(x_i - \bar{x}_i\right)}{\delta_i^2}$$

The rule for adjusting the parameter $\bar{x}_i$ is:

$$\bar{x}_i(t+1) = \bar{x}_i(t) + \eta p_t \tag{18}$$

where

$$p_t = -g_t + \beta_t p_{t-1}; \text{ and } g_t = \frac{\partial E}{\partial \bar{x}_i}$$

Similarly for δ_i the adjusting rule is calculated from:

$$\frac{\partial E}{\partial \delta_i} = \sum_{m \in C_{-i}^{(2)}} \left(\sum_{k \in S_{-m}^{(3)}} \left(\frac{\partial E}{\partial S_k^{(3)}} \frac{\partial S_k^{(3)}}{\partial S_m^{(2)}} \right) \frac{\partial S_m^{(2)}}{\partial S_i^{(1)}} \frac{\partial S_i^{(1)}}{\partial \delta_i} \right) \tag{19}$$

where

$$\frac{\partial S_i^{(1)}}{\partial \delta_i} = e^{-\frac{(x_i - \bar{x}_i)^2}{2\delta_i^2}} \frac{(x_i - \bar{x}_i)^2}{\delta_i^3}.$$

The rule for fixing the parameter δ_i is:

$$\delta_i(t+1) = \delta_i(t) + \eta p_t \tag{20}$$

where

$$p_t = -g_t + \beta_t p_{t-1}; \text{ and } g_t = \frac{\partial E}{\partial \delta_i}$$

For the nodes of the first layer that correspond to the variable Dist, the parameters a, b and c must be adjusted by using the same technique of conjugated gradients. Only one difference must be applied because a discrete activation function must be applied, so to find a direction to reduce the error, the Lagrange Interpolation is proposed. For this particular problem it is used:

$$P(x_i) = \frac{(x_i - 1)(x_i - 2)}{2} a - (x_i)(x_i - 2)b + \frac{(x_i)(x_i - 1)}{2} c \tag{21}$$

Once the polynomial is obtained, the adjusting rule for the parameter a can be calculated, as on the parameters $\bar{x}$ and δ from the formulae:

$$\frac{\partial E}{\partial a_i} = \sum_{m \in C_{-i}^{(2)}} \left(\sum_{k \in S_{-m}^{(3)}} \left(\frac{\partial E}{\partial S_k^{(3)}} \frac{\partial S_k^{(3)}}{\partial S_m^{(2)}} \right) \frac{\partial S_m^{(2)}}{\partial S_i^{(1)}} \frac{\partial S_i^{(1)}}{\partial a_i} \right) \tag{22}$$

where

$$\frac{\partial S_i^{(1)}}{\partial a_i} = \frac{(x_i - 1)(x_i - 2)}{2}.$$

The adjusting rule for the parameter a_i can be defined as follows:

$$a_i(t+1) = a_i(t) + \eta p_t \tag{23}$$

where

$$p_t = -g_t + \beta_t p_{t-1}; \quad \text{and} \quad g_t = \frac{\partial E}{\partial a_i}$$

Similarly, the adjusting rule for the b parameter is calculated from the formulae:

$$\frac{\partial E}{\partial b_i} = \sum_{m \in C_{-i}^{(2)}} \left(\sum_{k \in S_{-m}^{(3)}} \left(\frac{\partial E}{\partial S_k^{(3)}} \frac{\partial S_k^{(3)}}{\partial S_m^{(2)}} \right) \frac{\partial S_m^{(2)}}{\partial S_i^{(1)}} \frac{\partial S_i^{(1)}}{\partial b_i} \right) \tag{24}$$

where

$$\frac{\partial S_i^{(1)}}{\partial b_i} = -(x_i)(x_i - 2) \cdot$$

And the adjusting rule for the parameter b_i is defined by:

$$b_i(t+1) = b_i(t) + \eta p_t \tag{25}$$

where

$$p_t = -g_t + \beta_t p_{t-1}; \quad \text{and} \quad g_t = \frac{\partial E}{\partial b_i}$$

Finally the adjusting rule for the parameter c is calculated by:

$$\frac{\partial E}{\partial c_i} = \sum_{m \in C_i^{(2)}} \left(\sum_{k \in S_m^{(3)}} \left(\frac{\partial E}{\partial S_k^{(3)}} \frac{\partial S_k^{(3)}}{\partial S_m^{(2)}} \right) \frac{\partial S_m^{(2)}}{\partial S_i^{(1)}} \frac{\partial S_i^{(1)}}{\partial c_i} \right) \tag{26}$$

where

$$\frac{\partial S_i^{(1)}}{\partial c_i} = \frac{(x_i)(x_i - 1)}{2}.$$

Then the adjusting rule for the parameter c_i comes from:

$$c_i(t+1) = c_i(t) + \eta p_t \tag{27}$$

where

$$p_t = -g_t + \beta_t p_{t-1}; \text{ and } g_t = \frac{\partial E}{\partial c_i}$$

It can be observed from the characteristics of the interpolation and because it is a discrete function, that the error differential caused by the parameter variance is many times zero. This value brings itself a problem with the update value, because it becomes undetermined. When this happens, an updating value is proposed by default, $\beta = 1$. Due to this situation, a third version for the neural network is proposed. This version includes the updates made from the second version and some modifications of the calculation of the values for a, b and c. This is made by implementing the maximum descent method, and it is indicated as follows:

$$a(t+1) = a(t) + \eta(-\frac{\partial E}{\partial a}) \tag{28}$$

$$b(t+1) = b(t) + \eta(-\frac{\partial E}{\partial b}) \tag{29}$$

$$c(t+1) = c(t) + \eta(-\frac{\partial E}{\partial c}) \tag{30}$$

Where, the differentials are calculated by using the same formulas from the first and second version. In the next section, the obtained results from the performance of the three versions of the neural network are presented.

5 Obtained Results

Many tests were developed in order to find the most efficient of the three versions of the neural network. The tests were made on a local server by simulating 20 games of the TAC Classic, for both learning factors 0.01 and 0.001, with the three implemented

versions of the neural network. The main objective is to find the best behavior for the prediction of the hotel room prices. To make the tests, two agents from the TAC Agent Repository [6] were downloaded and installed to compete in the platform –UTTA06 and MerTACor. The agents and the server were executed on the local network.

The three described versions of the neural network were tested, each with different input variables and optimization techniques. The networks RN1 and RN2 have the three input values –Preference, Average Price and Distribution-, and uses Conjugated Gradient for optimization and learning. For the third network RN3, only the Distribution input values were tested with Maximal Descent Optimization technique.

By using the learning factor f = 0.01 with RN1, the average prediction error is calculated in 49.5024696, with RN2 it is 56.233856, and with RN3 the error was 53.3619775. Considering a learning factor f = 0.001 the Average Prediction Error for RN1 is 59.7812932, with RN2, 52.07654 and with RN3, it is 52.0490384.

6 Conclusions

In this paper, the output and input values for the problem of the prediction of the hotel room prices were defined. Twenty-seven rules were established in order to build a neural network of five layers. The learning techniques implemented on the neural network, by using discrete activation functions was also explained, and is considered as the real contribution of this paper.

By changing the controlled variables, three different versions of the neural network were developed, named RN1, RN2 and RN3. By analyzing the obtained results on the presented tests, it can be inferred that RN3 with a learning factor f = 0.001 offers the best behavior of all the implementations. Then, an agent with this network will be used to predict the prices on the room hotel auctions.

A heuristic to solve the assignation problem for bidding prices for tickets for the amusement shows has been designed, and for the assignment of the remaining tickets a differential equation has been proposed. At this moment, we are at the testing phase of the agent; we are currently analyzing how the maximum income is guaranteed by depending on the constructed tour packages.

References

1. Trading Agent Competiton, Game Description http://www.sics.se/tac/page.php?id=3
2. Sardinha J., Milidiú R., Paranhos P., Cunha P., Lucena C. *An Agent Based Architecture for Highly Competitive Electronic Markets*. Proceedings of the 18th Internanional FLAIRS Conference, pp. 326-331.
3. Toulis P., Kehagias D., Mitras P. *Mertacor: A Successful Autonomous Trading Agent*. AAMAS'06 (May 8-12, 2006, Hakodate, Hokkaido, Japan)
4. Wellman M., Reeves D., Lochner K., Suri R. *Searching for Walverine 2005*. IJCAI-05 Workshop on Trading Agent Design and Analysis, pp. 1-6, Nineteenth

International Joint Conference on Artificial Intelligence (30 July - 5 August, 2005, Edinburgh, Scotland).

5. He M., Jennings N., Prügel- Bennet A. *A Heuristic Bidding Strategy for Buying Multiple Goods in Multiple English Auctions.* ACM Transactions on Internet Technology, 2005.
6. Trading Agent Competition, TAC Agent Repository http://www.sics.se/tac/showagents.php

Reinforcement Learning based Neurocontrollers

Erik Cuevas[1,2], Daniel Zaldivar[1,2], Marco Perez[2] and Raúl Rojas[1]

[1] Freie Universität Berlin, Takustr. 9
14195 Berlin, Germany
{cuevas, zaldivar, rojas}@inf.fu-berlin.de
http://www.inf.fu-berlin.de
[2] Universidad de Guadalajara, CUCEI, Av. Revolución 1500,
44430 Guadalajara, Jalisco, Mexico
marcop@cucei.udg.mx

(Paper received on June 22, 2007, accepted on September 1, 2007)

Abstract. An important characteristic of a controller is the adaptation to dynamic changes in the system to be controlled; however most of the algorithms used consist on complex and computationally expensive structures. This paper presents a controller based on radial neural networks which is able to perform parameter adaptation as a response to changes on the system dynamics using a simple reinforcement learning rule. The approach has been tested on a typical non-linear benchmark plant: the steering ship control.

1 Introduction

Conventional controllers are only efficient where the system to be controlled (the plant) or rather the model of that system represented within the controller is characterized by constant parameters applicable at all operating points. And yet, most complex systems are characterized by parameters that vary with the system operating point, thus failing to meet the basic assumption just stated.

Neural networks have been successfully applied on identification and control of dynamical systems. They are also regarded as good approximation elements for modelling non-linear systems, with remarkable results on designing neurocontrollers [1]. Advantages seem evident as they do not require the plant's model as a proper selection on data from system's input and output might suffice.

There exist several neural architectures for controlling dynamic systems [2]. The design procedure includes two steps, first the identification of the plant dynamics and second the controller generation from the inverse approximation of operational data. Following this assumption, the new controller would have the ability to control the plant by constantly keeping the same dynamic characteristics, which is not sometimes the case.

Radial function neural networks may require more neurons than a classical feed-forward network. However, the weights on the overall structure hold a semantic meaning which in turn allows a customized update rule and therefore approaches each neu-

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 173-182

ron accordingly to its overall contribution [3], and therefore have been considered as an attractive proposal for online adaptation.

Considering the use of reinforcement learning in control engineering, the controller learns how to lead the plant through direct interaction by generating signals to the plant and evaluating their impact. The controller would therefore modify its parameters according to the relative success or failure of its actions [4]. Applying reinforcement learning to control implies that the controller learns how to stabilize the plant through a direct interaction. It applies input signals to the plant measuring the output to evaluate overall impact of the control signal. Hence the controller updates its parameters by evaluating the relative success of its actions [4].

This work presents a controller based on radial neural networks which is able to perform parameter adaptation as a response to changes on the system´s dynamics by using a simple reinforcement learning rule based on a reference model of the plant response. The approach has been tested on the steering ship control.

This work is organized as follows: section 2 present a brief description of the neural network theory, in particular to radial base networks. Section 3 describes main features on reinforcement learning while section 4 shows the ship steering problem which is used in demonstrations. Section 5 explains the reinforcement signal used in updating the parameters while Section 6 describes the radial-based neuro-controller architecture which lies on the foundations of this work. Section 7 presents the adaptation law applied to the fixed controller while Section 8 discusses some conclusions and future work.

2 Radial Base Neural Networks

Radial base neural networks (RBNNs) use functions whose output depends on the distance between the input and a value w considered as reference. Figure 1 shows a representation of such arrangement. RBNNs differ from other feedforward networks on the fact that the activation function receives as input the difference ($|\text{dist}|$) between the input vector and the weights instead of its arithmetic product.

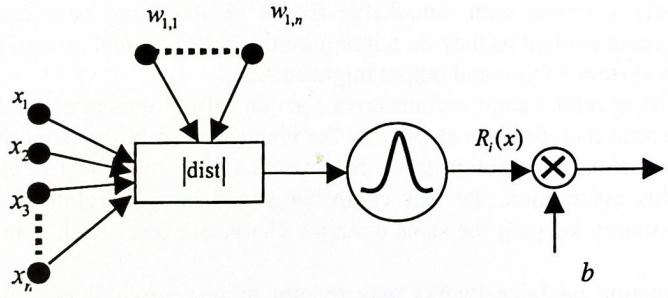


Fig. 1. Schematic representation of one radial base neuron.

The activation function normally employed is a Gaussian [6] as represented by the following equation:

$$R_i(x) = \exp\left(\frac{|x-w|}{(\sigma^i)^2}\right) \tag{1}$$

The network's output y is computed by the product $R_i(x)$ times b. The last parameter becomes important given that it can modify the sensitivity of function $R_i(x)$ in the network input. Hence, as long as the input vector **x** goes further away from the centre on $R_i(x)$ as represented by vector **w**, the value of $R_i(x)$ decreases.

3 Reinforcement Learning

In control engineering, appropriate actions required by the plant to keep the requirements are commonly unknown. In the case of non-linear systems, neural networks have shown good skills to be applied for identification by using the backpropagation learning rule. Unfortunately it also shows some drawbacks. It requires the network outputs in advance to training. Reinforcement Learning (RL) method holds a more convenient feature for the systems control problem. Instead of requiring an appropriate control action to learn from, it accepts a reward index to score its own actions, commonly known as the critic. Such element is able to define an acceptable or disappointing performance on following the required control strategy. The overall method resides in the middle of supervised and unsupervised algorithms.

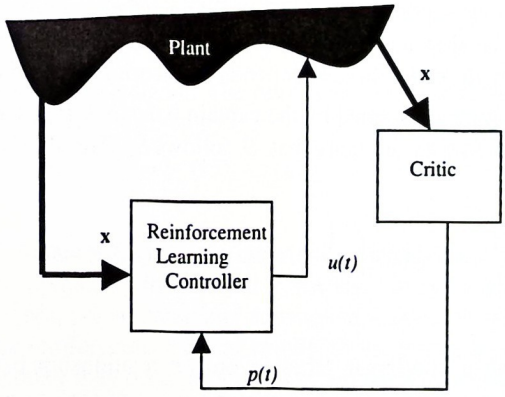


Fig. 2. Reinforcement Learning control scheme.

After receiving the system state **x** (Fig. 2), the learner receives reinforcement $p(t)$ from the environment notifying about the usefulness of its output $u(t)$. The main objec-

tive is therefore to maximize this reward signal over the time [7]. This can be achieved by trial and error training until the learner is able to discover those outputs with a maximum reward.

The main component within an RL based control scheme is therefore the critic signal and how it is processed to adjust the controller parameters. The solution proposed in this paper employs radial base neural networks and a critic signal named as J_R. In order to provide a reliable critic signal to represent how the plant must behave on real time, our implementation uses a reference model based approach.

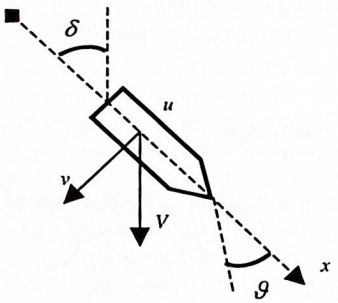


Fig. 3. The ship´s model.

4 The Plant Model

The testbed in this paper is the ship´s steering wheel control problem [8] as explained in Fig. 3. The ship moves forward in the x direction with a speed u, while ϑ refers the steering angle which in turn depends on steering wheel angle δ. Hence ϑ_r is the target direction angle as defined by the captain o desired trajectory. The objective is to develop a control strategy to assure that ϑ follows ϑ_r. The ship´s movements can be

$$\dddot{\vartheta}(t)+\left(\frac{1}{\tau_1}+\frac{1}{\tau_2}\right)\ddot{\vartheta}(t)+\left(\frac{1}{\tau_1\tau_2}\right)H(\dot{\vartheta}(t))=\frac{K}{\tau_1\tau_2}(\tau_3\dot{\delta}(t)+\delta(t)) \tag{2}$$

with $H(\dot{\vartheta}(t))$ being a non-linear function on the relationship between δ and ϑ on steady state. From a well-known test called the "spiral" [1], the function can be approximated by

$$H(\dot{\vartheta}(t))=\overline{a}\dot{\vartheta}^3(t)+\overline{b}\dot{\vartheta}(t) \tag{3}$$

with $\overline{a}$ and $\overline{b}$ being real valued and $\overline{a}$ always positive. This paper considers $\overline{a}=\overline{b}=1$ while δ is limited to ± 80 degrees as in [8]. The value of K and constants τ_i depends on the ship's speed u.

5 Reinforcement Signal based on the Reference Model

Commonly a reference model is used to score the desired performance on closed-loop. A fixed trajectory is generated from the reference model as to define how the plant must behave on each time instant. In such model, all performance indexes must therefore be clearly defined. If the reference model holds a very strict behaviour model in terms of performance, then the controller would never reach a satisfactory adaptation to it.

In general, the reference model may be continuous or discrete, linear or non-linear, time invariant or not. In this paper, the reference model which represents a correct response in time is a continuous expression as follows:

$$G(s)=\frac{1}{s+1}=\frac{Y_m(s)}{R(s)} \tag{4}$$

The reference model is discretized using as sample time T=0.1 seconds and its bilinear transform is defined as follows:

$$y_m(kT)=\frac{19}{21}y_m(kT-1)+\frac{1}{21}r(kT)+\frac{1}{21}r(kT-1) \tag{5}$$

With $r(kT)$ is the reference signal. The reinforcement signal employed in this work by the parameter update rule is given by:

$$J_R=y_m(kT)-y(kT) \tag{6}$$

with $y(kT)$ being the plant's output. It is important to provide one more feature in case the plant's output approaches the desired behaviour. It due case the reinforcement learning should be zero, i.e. no parameter adjustment is done. In real-time operation, if small values on the reinforcement signal results on several changes on the controller parameter set, instability may occur as a result of unnecessary storage of such updating orders. In order to avoid such problem in this work, we employed a threshold function that allows changes if only the reinforcement signal may be consider as admissible. Such function is defined as follows (considering α =0.005).

$$J_R = \begin{cases} J_R \text{ if } |J_R| \geq \alpha \\ 0 \quad \text{if } |J_R| < \alpha \end{cases} \tag{7}$$

6 The Neurocontroller

This section describes the most important features in the neurocontroller with respect to its own ability to modify its behaviour as a response to changes on the plant's dynamics. The radial base neural network controller follows the work of Passino in [5].

The controller seeks to regulate the ship's direction by using 2 inputs: steering error and its derivative. The network architecture is shown in Fig. 4.

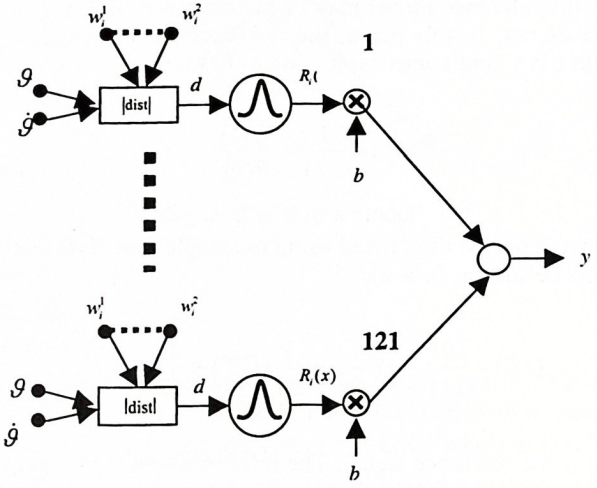


Fig. 4. The neural network architecture used as controller.

The network has 121 radial base neurons distributed among all input span for ϑ $\left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ (also known as **e** in this work), and for $\dot{\vartheta}$ (-0.01, 0.01) (also named as **c**). The radial base function parameters (Gaussians) are defined by the variants over the ϑ axis in both ways yielding

$$\sigma_\vartheta = 0.7\frac{\pi}{\sqrt{n_R}} \tag{8}$$

As for $\dot{\vartheta}$

$$\sigma_{\vartheta} = 0.7\frac{0.02}{\sqrt{n_R}} \tag{9}$$

In the expression above, n_R represents the linear neuron distribution. In this work such value is 11 with a total receptive field number of n_R^2=121. Using this data set, the overall input space is fully covered while allowing a smooth transition between neurons.

The values on vector $\mathbf{b} = [b_1 \quad b_2 \quad \cdots \quad b_{121}]$ are computed following the Passino's method on [5]. The training data set is obtained from simulation on the ship's dynamical model. Under this assumptions, the system was able to control the non-linear ship model (section 4), considering a speed of u=5 m/s. However, if a change on the plant's dynamics occurs, such as varying the ship's speed then the control rule is lost.

7 Reinforcement Learning based Controller Adaptation

There are several options to adjust the network weights using the reinforcement signal. This paper considers the update of vector $\mathbf{b} = [b_1 \quad b_2 \quad \cdots \quad b_{121}]$ which multiplies all 121 functions on the radial base (receptive fields).

The adaptation law on these parameters is defined by the following equation:

$$b_i(kT) = b_i(kT-1) + J_R(kT)R_i(\vartheta, \dot{\vartheta}) \tag{10}$$

The parameter b on neuron i is computed considering its previous value plus the result of the product between the reinforcement signal J_R and the radial base value R_i which triggered such neuron.

In order to test the performance of the adaptation law and the quality on the non-linear control law applied to the ship, the dynamics model is changed from u=5 m/s to 10 m/s. The fixed radial base neurocontroller (explained in section 6) is taken as reference and the vector **b** is changed according to the adaptation law presented in (10). The results are shown in Fig. 5.

It can be seen that the adaptation time is very small allowing testing the plant against to one dynamics change. In order to test the robustness on the controller adaptation, the effect of wind perturbations on the ship are considered. Assuming the wind is flowing on time intervals (as normally happens in the sea), the overall effect may be simulated by adding a sinusoidal signal to the steer angle (input to the plant) as follows:

$$2\left(\frac{\pi}{180}\right)\sin(2\pi(0.001)*t) \tag{11}$$

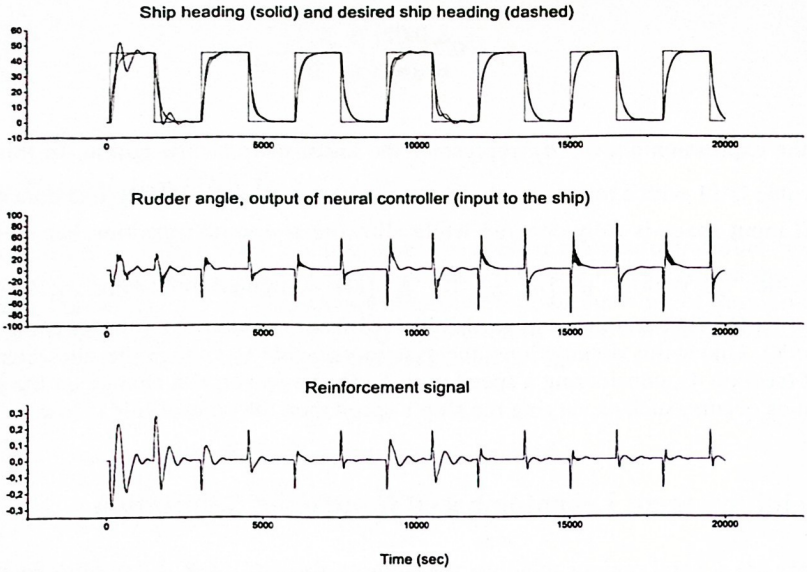


Fig. 5. Controller response using the adaptative controller and changing the dynamics on the model to u=10 m/s.

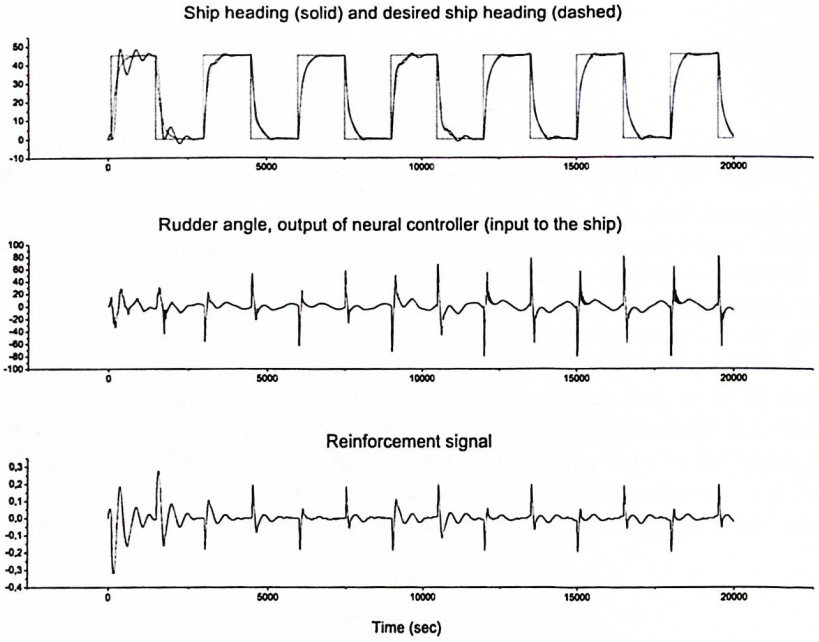


Fig. 6. Controller´s response one wind perturbation as defined in Eq. 11.

Figure 6 shows the controller's response to the perturbation. It can be seen how the controller is able to perform the required modifications to dismay the perturbation's effect which normally tend to generate instability. It can also be observed within the steady state analysis, how the wind effect can be modelled as a sinusoidal disturbance on the steer angle. Another important feature to consider in the controller adaptation comes from the fact of adding noise to the plant's output despite still keeping control of the plant. This test adds uniformly distributed and random noise to the output as shown in Fig. 7.

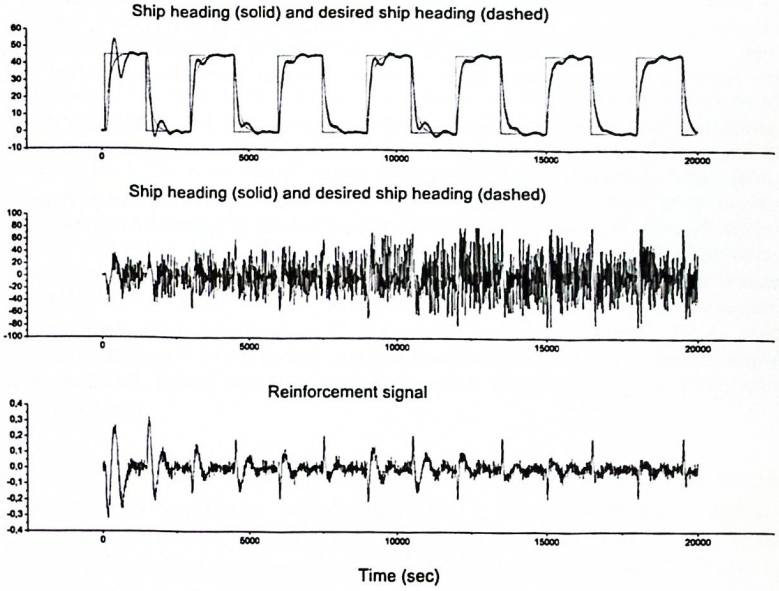


Fig. 7. Controller's response considering noise added to the output.

7 Conclusions

This paper presents a neural network based controller and a reinforcement learning update algorithm. The overall algorithm is able to exert control over a non-linear problem despite applying changes on the plant's dynamics. The reinforcement signal is computed from measurements on the plant's performance and its comparison to a simple reference model.

Despite it appropriately controls the plant, the neurocontroller initially shows some overshooting as a result of the parameter adaptation (Fig. 5). It takes about 1.2 seconds to define that no more changes are required in the controller's structure.

The updating algorithm based on reinforcement has shown an acceptable robustness, as it keeps the ship steering control despite perturbations shown by Fig. 6 or the noisy signals presented by Fig. 7.

References

1. Miller, W.T., Sutton, R.S. and Werbos, P.J., editors. Neural Networks for Control. The MIT Press, Cambridge, MA, 1991.
2. Narendra, K.S. and Parthasarathy, K., Identification and Control of Dynamical Systems using Neural Networks. IEEE Transactions on Neural Networks 1(1):4-27, 1990.
3. Chen, S., Billings, S.A., and Grant, P.M., Recursive hybrid algorithm for nonlinear system identification using radial basis function networks. International Journal of Control, 55(5):1051-1070, 1992.
4. Farrell, J.A., and Baker, W., Learning Control Systems. In P.J. Antsaklis and K.M Passino, editors, An Introduction to intelligent and Autonomous Control systems, pages 273-262. Kluwer academic publishers, Norwell, MA, 1993.
5. Passino, K.M., and Antsaklis, P.J., A System and Control Theoretic Perspective on Artificial Intelligence Planning Systems. International Journal of Applied Artificial Intelligence, 3:1-32,1989.
6. Sanner, R. M and Slotine, J. J. E. Gaussian Networks for Direct Adaptive Control. IEEE Transactions on Neural Networks, 3(6), 837-863.
7. Barto, A. G., Bradtke, S. J., and Singh, S.P. Learning to Act using Real-Time Dynamic Programming. Department of Computer Science, University of Massachusetts.
8. Amström K. J. and Wittenmark. Adaptive Control. Addison-Wesley, Reading, MA. 1995.

Direct Adaptive Vector Neural Control of a Three Phase Induction Motor

Ieroham S. Baruch, Carlos R. Mariaca-Gaspar, and Irving P. de la Cruz

CINVESTAV-IPN, Department of Automatic Control, Av. IPN No 2508,
Col. Zacatenco, A.P. 14-740, 07360 Mexico D.F., Mexico
{baruch, cmariaca, idelacruz}@ctrl.cinvestav.mx

(Paper received on June 07, 2007, accepted on September 1, 2007)

Abstract. The paper proposed a complete neural solution to the direct vector control of three-phase induction motor including real-time trained neural controllers for velocity, flux and torque, which permitted the speed up reaction to the variable load. The basic equations and elements of the direct field oriented control scheme are given. The control scheme is realized by nine feedforward and recurrent neural networks learned by Levenberg-Marquardt or real-time BP algorithms with data taken by PI-control simulations. The graphical results of modelling shows a better performance of the NN control system with respect to the PI controlled system realizing the same general control scheme.

1 Introduction

The application of Neural Networks (NN) for identification and control of electrical drives became very popular in last decade. In [1], a multilayer feedforward neural network is applied for a DC motor drive high performance control. In [2], a recurrent neural network is applied for identification and adaptive control of a DC motor drive mechanical system. In the last decade a great boost is made in the area of induction motor drive control. The induction machine of cage type is most commonly used in adjustable speed AC drive systems [3]. The control of AC machines is considerably more complex than that of DC machines. The complexity arises because of the variable-frequency power supply, the AC signals processing, and the complex dynamics of the AC machine [3], [4]. In the vector or Field-Oriented Control (FOC) methods, an AC machine is controlled like a separately excited DC machine, where the active (torque) and the reactive (field) current components are orthogonal and mutually decoupled so they could be controlled independently, [3]-[7]. There exist two methods for PWM inverter current control – direct and indirect vector control, [3]. This paper applied the direct control method, where direct AC motor measurements are used for field orientation and control. There are several papers of NN application for AC motor drive direct vector control. In [8] a feedforward NN is used for vector PW modulation, resulting in a faster response. In [9] an ADALINE NN is used for cancellation of the integration DC component during the flux estimation. In [10] a fuzzy-neural uncertainty observer is integrated in a FOC system, using an estimation of the rotor time constant. In [11] an Artificial NN is used for fast estimation of the angle used in a FOC

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 183-192

system. In [12] a flux and torque robust NN observer is implemented in a FOC system. In [13], an ADALINE-based-filter and angular-velocity-observer are used in a FOC system. In [14], the authors proposed a NN velocity observer used in FOC high performance system for an induction motor drive. In [15] a Feedforward-NN (FFNN)-based estimator of the feedback signals is used for induction motor drive FOC system. The paper [16] proposed two NN-based methods for FOC of induction motors. The first one used a NN flux observer in a direct FOC. The second one used a NN for flux and torque decoupling in an indirect FOC. The results and particular solutions obtained in the referenced papers showed that the application of NN offers a fast and improved alternative of the classical FOC schemes, [17]. The present paper proposed a neural solution of a direct FOC. The system achieved adaptation to a variable load applying real-time learned neural controllers of IM velocity, flux, and torque. In our early work, [17], the phase (a,b,c), the (q,d,0) model, and the coordinate transformation between them has been completely described, so here we may skip those parts.

2 Field Orientation Conditions and Flux Estimation

The flux and torque decoupling needs to transform the stator flux, current and voltage vectors from (a, b, c) reference frame into (q-d,s) reference frame and than to stationary and synchronous reference frames, [17]. In the next equations, the following notation is used: v –voltage, i-current, λ-flux, r-resistance, L-inductance, ω-velocity; the sub-indices are r- rotor, s-stator, q, d- components of the (q, d, 0) model; the upper index s means stator reference frame and e means synchronous reference frame; the prime means relative rotor to stator value. The Fig. 1a illustrates the current and voltage vector representations in stator and rotor synchronous frames and also the magnetic field orientation, where the rotor flux vector is equal to the d-component of the flux vector, represented in a synchronous reference frame ($\lambda'^{e}_{dr}=\lambda_r$), which is aligned with the d-component of the current in this frame. For more clarity, the current and flux orientation in the synchronous reference frame are shown on Fig. 1b. So, the field orientation conditions are:

$$\lambda'^{e}_{qr} = 0; p\lambda'^{e}_{qr} = 0; \lambda_r = \lambda'^{e}_{dr} \tag{1}$$

Taking into account that the rotor windings are shortcut, (the rotor voltage is zero), also the given up field orientation conditions, and the (q, d, 0) model, [17], we could write:

$$0 = r'_r i'^{e}_{qr} + (\omega_e - \omega_r)\lambda'^{e}_{dr}; 0 = r'_r i'^{e}_{dr} + p\lambda'^{e}_{dr} \tag{2}$$

Using the (q, d, 0) model, [17], for the q-component of the rotor flux, it is obtained:

$$\lambda'^{e}_{qr} = L_m i^{e}_{qs} + L'_r i'^{e}_{qr} = 0; L'_r = L'_{lr} + L_m; i'^{e}_{qr} = -(L_m / L'_r) i^{e}_{qs}; \tag{3}$$

Using (1) and (3), the torque equation [17] could obtain the form:

$$T_{em} = \frac{3}{2}\frac{P}{2}\frac{L_m}{L_r'}\lambda_{dr}'^{e} i_{qs}^{e} \tag{4}$$

The equation (4) shows that if the flux of the rotor is maintained constant, so the torque could be controlled by the q-component of the stator current in synchronous reference frame. From the second equation of (2), taking into account (3) it is easy to obtain the slipping angular velocity as:

$$\omega_e - \omega_r = (r_r' L_m / L_r')(i_{qs}^{e} / \lambda_{dr}'^{e}) \tag{5}$$

The final equations (3), (4), (5) gives us the necessary basis for a direct decoupled field oriented (vector) control of the AC motor drive, where (see Fig. 1b) the q- component of the stator current produced torque and its d-component produced flux. Following [17], we could write:

$$\lambda_{qs}^{s} = (1/p)\left(v_{qs}^{s} - r_s i_{qs}^{s}\right); \lambda_{ds}^{s} = (1/p)\left(v_{ds}^{s} - r_s i_{ds}^{s}\right) \tag{6}$$

$$i_{qr}'^{s} = (\lambda_{qs}^{s} - L_s i_{qs}^{s}) / L_m; i_{dr}'^{s} = (\lambda_{ds}^{s} - L_s i_{ds}^{s}) / L_m \tag{7}$$

$$\lambda_{qr}'^{s} = (L_r' / L_m)\left(\lambda_{qs}^{s} - L_s' i_{qs}^{s}\right); \lambda_{dr}'^{s} = (L_r' / L_m)\left(\lambda_{ds}^{s} - L_s' i_{ds}^{s}\right); L_s' = [L_s - (L_m^2 / L_r')] \tag{8}$$

Now it is easy to compute the angle needed for field orientation, the rotor flux, and the *sin, cos* - functions of this angle, needed for flux control, torque estimation, and coordinate transformations, which are:

$$\lambda_r' = \sqrt{(\lambda_{qr}'^{s})^2 + (\lambda_{dr}'^{s})^2}; sin\rho = \lambda_{qr}'^{s} / \lambda_r'; \cos\rho = \lambda_{dr}'^{s} / \lambda_r' \tag{9}$$

$$T_{em} = \frac{3}{2}\frac{P}{2}\frac{L_m}{L_r'}\lambda_r' i_{qs}^{e} \tag{10}$$

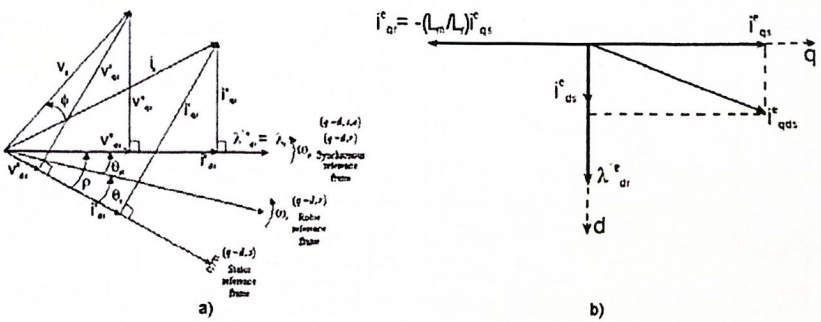


Fig. 1. Vector diagrams of the stator current, voltage and the rotor flux. a) The current and voltage vector representations in stator and in rotor synchronous reference frames. b) The stator current and the rotor flux vector representations in synchronous reference frame

3 General Control Scheme and NN Realization of the IM Control

A general block diagram of the direct vector control of the Induction Motor drive is given on Fig. 2a. The direct control scheme contains three principal blocks. They are: G1, G2, G3 – blocks of PI controllers; block of coordinate (abc) to (q-d,s,e) transformation, [17]; block of vector estimation, performing the field orientation and the torque, flux and angle computations (see equations (9), (10)); block of inverse (q-d,s,e) to (a,b,c) transformation; block of the converter machine system and induction motor. The block of the converter machine system contains a current three phase hysteresis controller; a three phase bridge ASCI DC-AC current fed inverter; an induction motor model; a model of the whole mechanical system driven by the IM ($(2/P)J(d\omega/dt)=T_{em}-T_L$, where J is the moment of inertia, T_L is the load torque). The block of vector estimation performed rather complicated computations. The Fig. 2b illustrates the flux and angle estimation for field orientation, computing (6), (8), (9). The rotor flux computations block (see Fig. 2b) performs computations given by (6), (8), illustrated by the Fig. 2c. The rotor flux, the angle, and the *sin, cos* -functions computations are given by equation (9). The torque estimation is computed by equation (10).

3.1 Neural Network Realization of the Control Scheme

The simplified block-diagram of the direct neural vector control system, given on Fig. 2a is realized by nine FFNNs. We will describe in brief the function, the topology and the learning of each FFNN. The main contribution here is the introduction of the neural P/PI velocity, flux and torque controllers which are capable to adapt the control system to load changes.

The FFNN1. The first NN1 is an angular velocity neural PI controller with two inputs (the velocity error, and the total sum of velocity errors) and one output (the torque set point). The weights learning is done in real – time using the Backpropagation (BP) algorithm. The FFNN1 function is given by the following equation:

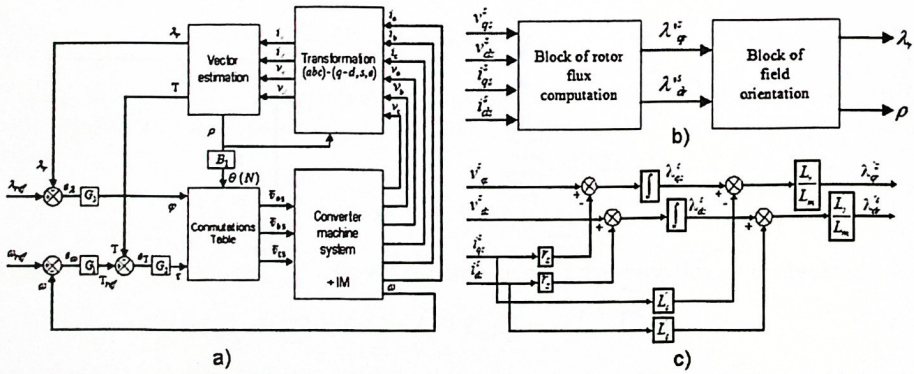


Fig. 2. Block diagrams. a) General BD of a direct IM vector control. b) BD of the vector estimation computations. c) BD of the flux estimation computations

$$T^*(k) = \varphi[g_p(k)e_{vel} + g_i(k)e_{vel}^{sum}(k)] \tag{11}$$

Where: g_p and g_i are proportional and integral FFNN1 weights; φ is a *tanh* activation function; e_{vel} is a velocity error; T^* is the torque set point – output of the FFNN1. The integration sum of errors is:

$$e_{vel}^{sum}(k) = \sum_{k=0}^{n} e_{vel}(k) \tag{12}$$

Where n is the total number of iterations. The BP algorithm for this FFNN1 is:

$$\begin{aligned} g_p(k+1) &= g_p(k) + \eta e_{vel}(k)[1-(T^*(k))^2]e_{vel}(k) \\ g_i(k+1) &= g_i(k) + \eta e_{vel}(k)[1-(T^*(k))^2]e_{vel}^{sum}(k) \end{aligned} \tag{13}$$

The FFNN2. The second FFNN2 is a torque neural P controller with one input and one output (the torque error and the stator q-current set point). The function and the real-time BP learning of this FFNN2 are given by:

$$i_{qs}^{e^*}(k) = \phi[g_p(k)e_T(k)] \tag{14}$$

$$g_p(k+1) = g_p(k) + \eta e_T(k)[1-(i_{qs}^{e^*}(k))^2]e_T(k) \tag{15}$$

Where: g_p is a proportional weight; φ is a *tanh* activation function; e_T is a torque error; η is a learning rate parameter; i_{qs}^{e}* is a current set point - output of FFNN2.

The FFNN3. The third FFNN3 is a flux neural PI controller with two inputs and one output (the flux error and its sum, and the stator d-current set point). The function and the real-time BP learning of this NN3 are given by:

$$i_{ds}^{e^*}(k) = \varphi[g_p(k)e_{flux} + g_i(k)e_{flux}^{sum}(k)] \tag{16}$$

$$\begin{aligned} g_p(k+1) &= g_p(k) + \eta e_{flux}(k)[1-(i_{ds}^{e^*}(k))^2]e_{flux}(k) \\ g_i(k+1) &= g_i(k) + \eta e_{flux}(k)[1-(i_{ds}^{e^*}(k))^2]e_{flux}^{sum}(k) \end{aligned} \tag{17}$$

Where: g_p and g_i are proportional and integral FFNN3 weights; φ is a *tanh* activation function; e_{flux} is a flux error; η is a learning rate parameter; i_{ds}^{e}* is a current set point - output of FFNN3. The integration sum of errors during n iterations is:

$$e_{flux}^{sum}(k) = \sum_{k=0}^{n} e_{flux}(k) \tag{18}$$

The FFNN4. The fourth FFNN4 is a torque off-line trained neural estimator (realizing (10) equation computation) which has two inputs and one output (the rotor flux, the stator q-current, and the estimated torque). The topology of this FFNN4 is (2-10-1).

The FFNN5. The fifth FFNN5 performed a stator current (a,b,c) to (q-d,s,e) transformation, [17]. The FFNN5 topology has five inputs (three i_{as}, i_{bs}, i_{cs} –stator currents; $\sin\varphi$, $\cos\varphi$), two outputs (i_{qs}^{e}, i_{ds}^{e} – stator currents) and two hidden layers of 30 and 20 neurons each (5-30-20-2).

The FFNN6. The sixth FFNN6 performed an inverse stator current (q-d,s,e) to (a,b,c) transformation using the transpose of the transformation matrix, [17]. The FFNN6 topology is (4-30-10-3) (four inputs -two i_{qs}^{e}, i_{ds}^{e} –stator currents; $sin\theta$, $cos\theta$; three outputs- i_{as}, i_{bs}, i_{cs} – stator currents; two hidden layers of 30 and 10 neurons).

The FFNN7. The seventh FFNN7 performed rotor flux estimation using equation (9). The rotor (q-d,r) flux components $\lambda_{qs}^{s}, \lambda_{ds}^{s}$ are previously computed using equation (6) (see Fig. 2c), and they are inputs of FFNN7. The other two inputs are the stator currents: i_{qs}^{s}, i_{ds}^{s}. The FFNN7 output is the rotor flux: $\lambda_{r}^{'}$. The NN7 topology is (4-30-10-1).

The FFNN8 and FFNN9. The FFNN8, and FFNN9 are similar to FFNN7 and performed separately the q and d rotor flux components estimation using equations (8). The FFNN8, FFNN9 topologies are: (2-10-5-2). The values of $sin\theta$, $cos\theta$, (9), needed for the coordinate transformations are obtained dividing the outputs of NN8, NN9 by the output of NN7.

All the FFNN4-9 are learned by 2500 input-output patterns (half period) and generalized by another 2500 ones (the other half period). The FFNN4-9 learning is off-line, applying the Levenberg-Marquardt algorithm [18], [19] during 61, 29, 32, 35, 47 and 49 epochs of learning, respectively. The final value of the MSE reached during the learning is of 10^{-10} for all the FFNN4-9.

4 Graphical Results of the Control System Modeling

The parameters of the IM used in the control system modelling are: power- 20Hp; nominal velocity – N = 1800 Rev.pm; pole number P = 4; voltage- 220 volts; nominal current – 75 A; phase number 3; nominal frequency 60 Hz; stator resistance r_s = 0.1062 Ohms; rotor resistance referenced to stator r_r'= 0.0764 Ohms; stator inductance L_s = 0.5689. 10^{-3} Henry; rotor inductance referenced to stator L_r' = 0.5689. 10^{-3} Henry; magnetizing inductance L_m = 15.4749. 10^{-3} Henry; moment of inertia J = 2.8 kg.m^2. The control system modeling is done changing the load torque in different moment of time. The Fig. 3a, b showed the angular velocity set point vs. the IM angular velocity in the general case of velocity control and particularly with load torque changes (PI and NN control). The results show that the angular velocity control system has a fast speed up response and satisfactory behaviour in the case of load change. The Fig. 4 a, b showed the flux graphics of control system with hysteresis control applying the PI control scheme and NNs. The results show a faster and better response of the neural system which tried to maintain the flux constant in the case of load changes. The Fig. 5 a, b; Fig. 6 a, b; Fig. 7 a, b, c, d show the torque and current graphics with hysteresis control in the same cases and load changes.

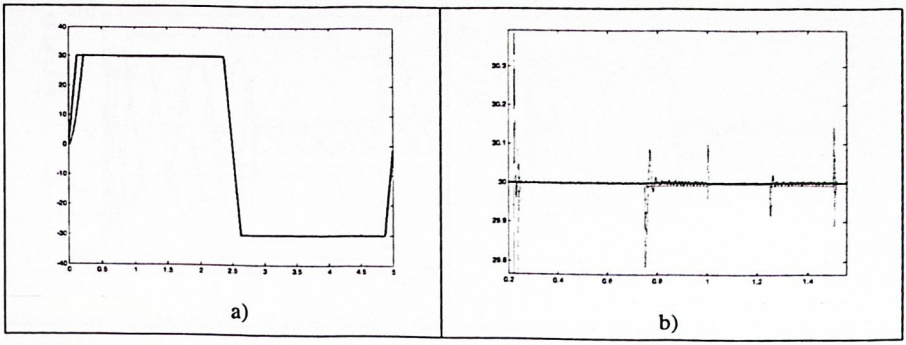


Fig. 3. Graphical results of the IM velocity control. a) General graphics of the angular velocity control; b) Graphical results of angular velocity control with load changes

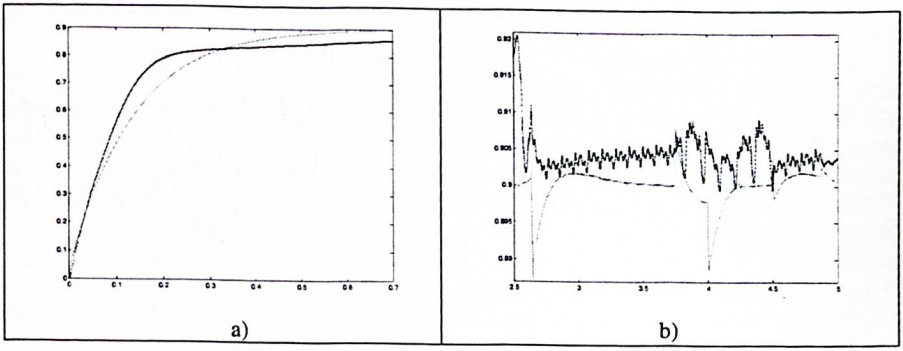


Fig. 4. Graphical results of the IM flux control. a) Graphics of the flux classical control vs. flux neural control; b) Graphics of both (classical vs. NN) flux control with load changes

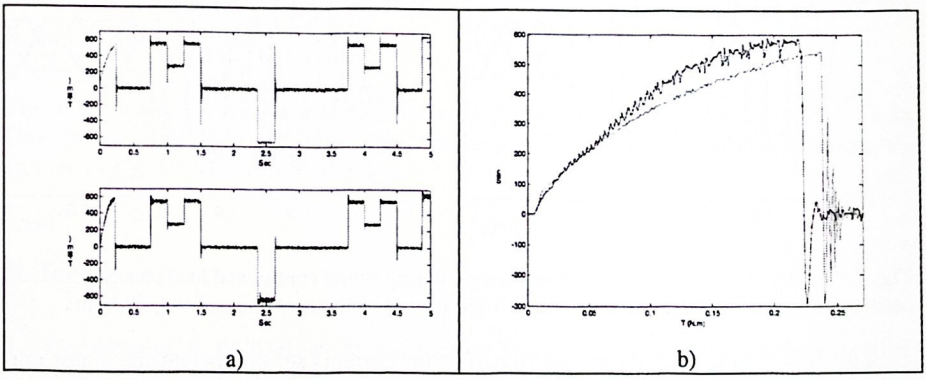


Fig. 5. Graphics of the torque control with load changes. a) Graphics of the torque classical and neural control; b) Graphics of both torque control (PI control vs. neural control) in the IM start

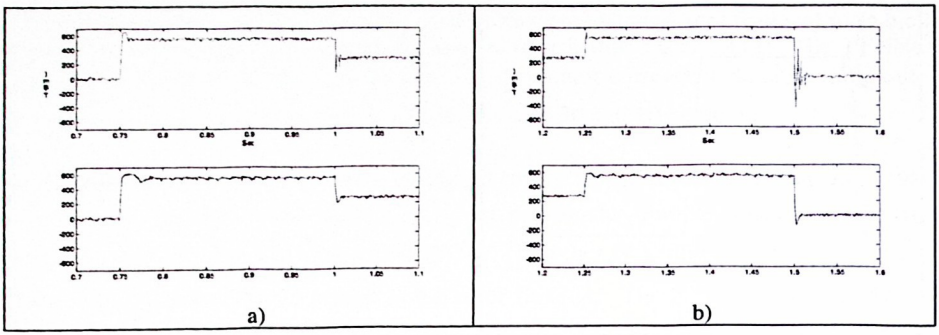


Fig. 6. Detailed graphics of the torque control using both control schemes and load changes. a) Processes from 0.7 sec. to 1.1 sec; b) Processes from 1.2 sec. to 1.6 sec

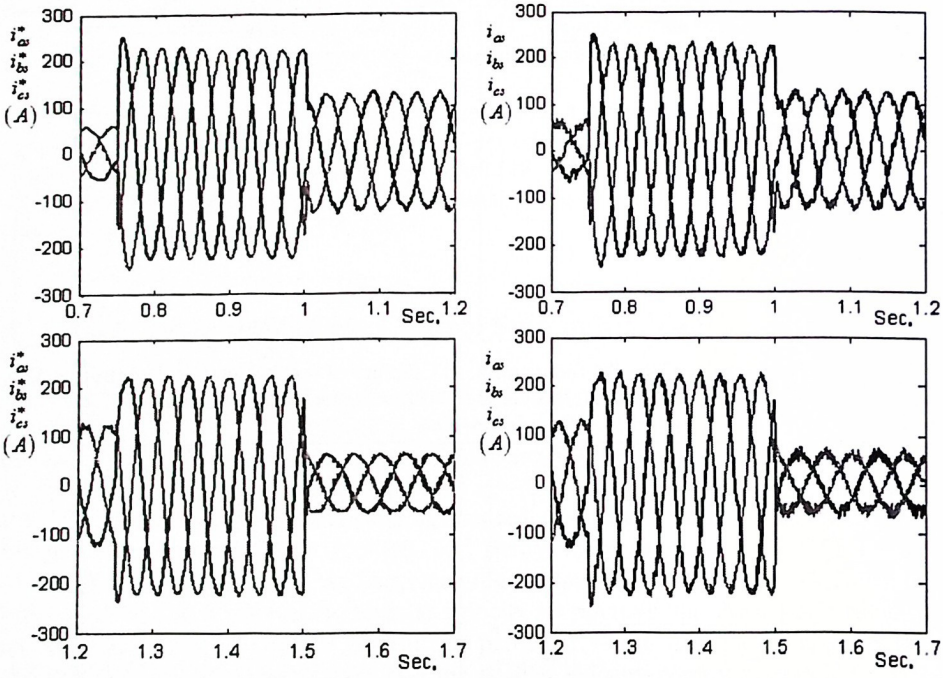


Fig. 7. Graphical results of (a,b,c) stator currents during neural control and load changes for periods of time of (0.7-1.2 sec.) and (1.2-1.7 sec.). a), c) Current set points; b), d) Currents

The Fig. 7 a, b, c, d shows the (a,b,c) stator currents set points and the stator currents of hysteresis controlled system using neural control schemes in load changes conditions for different time intervals. The Fig. 8 a, b shows the same stator current set points and currents during the start of the IM. The results show a good performance of the neural control system at all.

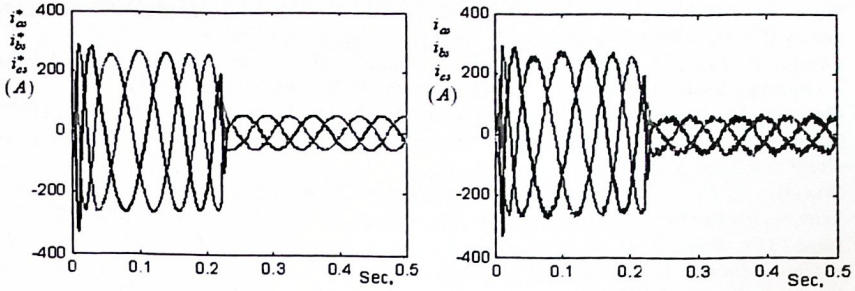


Fig. 8. Graphical Results of (a,b,c) Stator Currents During the Start of the IM. a) Current Set Points; b) Currents

5 Conclusions

The paper proposed a complete neural solution to the direct vector control of three phase induction motor including real-time trained neural controllers for velocity, flux and torque, which permitted the speed up reaction to the variable load. The basic equations and elements of the direct FOC scheme are given. The control scheme is realized by nine feedforward neural networks learned with data taken by PI-control simulations. The NN PI or P adaptive neural controllers are learned on-line using the BP algorithm. The complementary blocks which realized coordinate and computational operations are learned off-line using the Levenberg-Marquardt algorithm with a 10^{-10} set up error precision. The graphical results of modelling shows a better performance of the adaptive NN control system with respect to the PI controlled system realizing the same computational control scheme with variable load.

Acknowledgements

The PhD student Carlos-Roman Mariaca-Gaspar and MS student Irving-Pavel de la Cruz A. are thankful to CONACYT, Mexico for the scholarship received during his studies in CINVESTAV-IPN, Mexico.

References

1. Weerasooriya, S., El-Sharkawi, M. A.: Adaptive Tracking Control for High Performance DC Drives. IEEE Trans. on Energy Conversion, 4 (1991) 182-201.
2. Baruch, I. S., Flores, J. M., Nava, F., Ramirez, I. R., Nenkova, B.: An Advanced Neural Network Topology and Learning, Applied for Identification and Control of a D.C. Motor. In: Sgurev, V., Jotsov, V. (eds.): Proc. of the 1-st Int. IEEE Symposium on Intelligent Systems. IEEE, ISBN 0-7803-7601-3, Varna, Bulgaria, (2002) 289-295.

3. Bose, B. K.: Power Electronics and AC Drives. Prentice-Hall , Englewood Cliffs, New Jersey 07632, (1986) 264-291.
4. Ortega, R., Loría, A., Nicklasson, P., Sira – Ramírez, H.: Passivity – Based Control of Euler – Lagrange Systems. Springer – Verlag, London, Berlin, Heidelberg, (1998).
5. Ong, Ch. M.: Dynamic Simulation of Electric Machinery. Prentice Hall, New York, (1998).
6. Novotny, D. W., Lipo, T. A.: Vector Control and Dynamics of AC Drives. Oxford University Press, New York, (1996).
7. Woodley, K.M., Li, H., Foo, S.Y.: Neural Network Modeling of Torque Estimation and d-q Transformation for Induction Machine. Engng Appl. of Artif. Intell., 18 (1) (2005) 57-63.
8. Pinto, J.O., Bose, B.K., da Silva, L. E. Borges: A Stator – Flux – Oriented Vector – Controlled Induction Motor Drive With Space – Vector PWM and Flux – Vector Synthesis by Neural Networks. IEEE Trans. on Industry Applications, 37 (2001) 1308 – 1318.
9. Cirrincione, M., Pucci, M., Capolino, G.: A New Adaptive Integration Methodology for Estimating Flux in Induction Machine Drives. IEEE Trans. on PE, 19 (1) (2004) 25-34.
10. Lin, F.J., Wai, R.J., Lin, C.H., Liu, D.C.: Decoupled Stator-Flux-Oriented Induction Motor Drive with Fuzzy Neural Network Uncertainty Observer. IEEE Trans. on Industrial Electronics, 47 (2) (2000) 356-367.
11. Keerthipala, W.L., Duggal, B.R., Chun, M.H.: Implementation of Field – Oriented Control of Induction Motors using Neural Networks Observers. In: Proc. IEEE International Conference on Neural Networks, 3 (1996) 1795 – 1800.
12. Marino, P., Milano, M., Vasca, F.: Robust Neural Network Observer for Induction Motor Control. In: Proc. 28-th Annual IEEE PE Specialists Conference, 1 (1997) 699-705.
13. Cirrincione, M., Pucci, M., Capolino, G.: A New TLS Based MRAS Speed Estimation UIT Adaptive Integration for High-Performance Induction Machine Drive. IEEE Trans. on Industry Applications, 40 (4) (2004) 1116-1137.
14. Cirrincione, M., Pucci, M., Capolino, G.: An MRAS Based Sensorless High Performance Induction Motor Drive with a Predictive Adaptive Model. IEEE Trans. on Industrial Electronics, 52 (2) (2005) 532-551.
15. Simoes, M. G., Bose, B. K.: Neural Network Based Estimation of Feedback Signals for a Vector Controlled Induction Motor Drive. IEEE Trans. on Ind. Appl., 31 (1995) 620 – 629.
16. Ba – Razzouk, A., Cheriti, A., Olivier, G., Sicard, P.: Field – Oriented Control of Induction Motors Using Neural Network Decouplers. IEEE Trans. on P. Elec., 12 (1997) 752 – 763.
17. Baruch, I.S., Mariaca-Gaspar, C.R., De la Cruz, I.P.: A Direct Torque Vector Neural Control of a Three Phase Induction Motor. Research in Computer Science, ISSN 1870-4069. Special Issue on Neural Networks and Associative Memories (Sossa-Azuela, J.H., Baron-Fernandez, R. (Eds)), 21 (2006) 131-140.
18. Hagan, M. T., Menhaj, M. B.: Training Feedforward Networks with the Marquardt Algorithm. IEEE Trans. on Neural Networks, 5 (1994) 989-993.
19. Demuth, H., Beale, M.: Neural Network Toolbox User's Guide, version 4, The Math Works, Inc. COPYRIGHT, (1992 – 2002).

Fault Diagnosis in Power Transmission Networks Using Bayesian Networks

Luis E. Garza[1] and Geovanna Ruffo[2]

Mechatronics Department[1], Graduate Program on Automation[2]
Monterrey Institute of Technology
7th Building, 3rd floor, 2501 Garza Sada Avenue
Monterrey, NL, 64849, Mexico
{legarza,a00773991}@itesm.mx

(Paper received on June 11, 2007, accepted on September 1, 2007)

Abstract. In this paper we present an approach to detect and diagnose single or multiple faults in power transmission networks using Bayesian Networks (BNs). A BN model of the power system is generated by taking in account the relationships existing between the nodes in the network and the primary and secondary protection elements. The diagnostic system is implemented by using two different inference algorithms: a first-order logic theorem prover (used by the Independent Choice Logic framework), and an approximate inference method based on rejection sampling and likelihood weighting sampling. The input to the diagnostic system is a set of discrete alarms coming from the status of breakers at the moment of failure. The output of the system is a set of explanations to the observed symptoms. In both cases, the theorem prover and the approximate method, the main interest lies on the most likely explanation. The approach is tested by diagnosing faults in a simulated electrical power network with 12 nodes and 32 protection breakers.

1 Introduction

The purpose of monitoring a process or system operation is to reduce the occurrence of sudden or dangerous shutdowns, equipment damage, and personal accidents and to assist in the operation of the maintenance program. In any of these cases, appropriate and timely action becomes crucial, hence the importance of developing algorithms that gives an optimal trade-off between finding an exact solution and finding it quickly.

It is no surprise that the problem of process monitoring and fault diagnosis has become an important area of research in the Artificial Intelligence community. In general terms, a monitoring system for process operation should consider not only early detection and diagnosis, but also robustness, adaptability and reasonable storage and computational resources.

As the need to solve more complex real world decision problems has increased, the adequate treatment of uncertainty has become fundamental. This is especially true in large, highly interconnected systems such as power transmission networks, where for a given fault scenario, hundreds or thousands of alarms are

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 193-202

generated. The complexity and high degree of interconnection present in electrical power networks, can lead to an overwhelming array of alarms and status messages being generated as a result of a disturbance. This can have a negative impact on the speed with which operators can respond to a contingency, and therefore it becomes necessary to use automated tools that can help the operator to speed up the resolution process.
In the last decade, different AI algorithms have been proposed for solving the diagnosis problem in power networks: alarm processing aids have relied on the use of Expert Systems [4,5], Neural Networks [10], Fuzzy Logic [1] or Petri Nets [9]. More recently, the need to develop more powerful approaches has motivated the development of systems based on Bayesian Networks [11,12] that can deal very efficiently with uncertainty inherent to power systems. BNs are powerful graphical probabilistic models that encode in a compact way multivariate probability density functions.
In this paper we describe an approach to automate fault diagnosis in electrical power transmission networks using a discrete Bayesian network formalism. In order to perform the inference within the BN model, two different approaches are used: a theorem prover for a probabilistic first-order logic framework, and an approximate method based on rejection sampling and likelihood weighting. The general idea is to explore the possibility of having a diagnostic system able to perform online diagnosis. The input to the diagnostic system is a set of discrete alarms representing the status of protection breakers. The system generates explanations consistent with the discrete observations: each explanation contains the hypothesized nodes in a faulty state. We show results from experiments in a simulated power network with 12 nodes and 32 protection breakers.
The paper is organized as follows: Section 1 provided a general introduction. In section 2 we describe the necessary background regarding BNs and inference methods. In Section 3 we develop a case study. In section 4 we analyze the related work, and finally section 5 sets forth our conclusions.

2 Background

2.1 Bayesian Networks

BNs are directed acyclic graph (DAG) in which nodes represent random variables and arcs determine the probabilistic information needed to specify the joint probability distribution of all network variables, as shown in figure 1. In this BN model, every node has a set of discrete states and the functional relationships between power network nodes and protection breakers nodes are specified by the arcs. To specify the probability distribution of a BN, one must provide also the prior probabilities of all root nodes, and the conditional probabilities of the rest of the nodes given all the possible combinations of their direct predecessors (see for instance table 1).

A BN constitutes an efficient representation of a joint probability distribution over domain variables. Formally, it can be stated in this way: given a set of random variables $X_1, X_2, \ldots X_n$, each of which has a domain $Dom[X_i]$ of

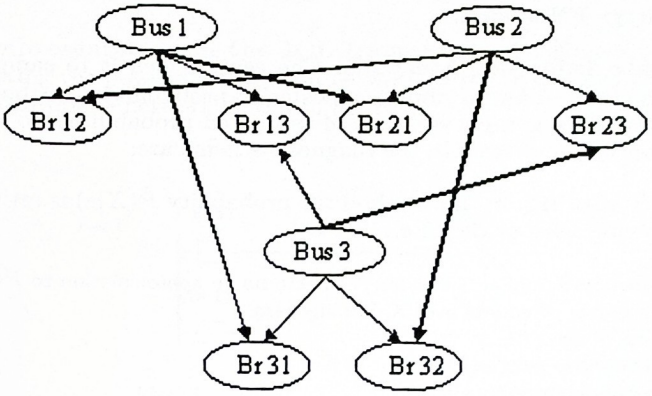


Fig. 1. Bayes network representation of the power network

Table 1. Probability distribution for breaker *Br12* node in the power network

Bus 1	**Ok**		**Fault**	
Bus 2	**Ok**	**Fault**	**Ok**	**Fault**
Normal	*1*	*0.9*	*0*	*0*
Open	*0*	*0.1*	*0.95*	*0.95*
Fail	*0*	*0*	*0.05*	*0.05*

possible values, then the full joint distribution of a BN is specified by the *chain rule*:

$$p(X_1, \ldots, X_n) = \prod_{i=1}^{n} p(X_i | Parents(X_i)). \tag{1}$$

BNs allow causal representations of observation dependencies as well as efficient algorithms for probabilistic inference; as a result, they are useful in modeling situations in which causality plays an important role but there is incomplete information. In these cases, BN's provide a potentially useful probabilistic representation. To perform inference over **BN**s, there are different methods such as: exact methods, first-order logic theorem provers and approximate methods. Exact methods allow an accurate calculation of probabilities, but are not adequate for large **BN** models due to excessive computing time. Theorem provers

are adequate when **BN** models are translated to first-order logic as in the Independent Choice Logic (**ICL**) framework [7]. Theorem provers have been used in the past with large **BN** models [2]. Approximate methods take samples from BNs considering the existing evidence. These methods are more adequate to apply over large BN models.

Approximate Inference Methods. The central idea is to sample the **BN** model N times in order to obtain a posterior probability distribution $\hat{P}$ and show whether there is a convergence of estimated probabilities to actual ones. The sampling methods used in the diagnosis scheme are:

Rejection Sampling. In this method the probability $\hat{P}(X|e)$ is estimated from samples agreeing with evidence e.

function Rejection Sampling (X, e, bn, N) **returns** an approximation to $P(X|e)$
  N[X] $\leftarrow$ a vector of counts over X, initially zero.
  for $j = 1$ to N do
    $x \leftarrow PriorSample(bn)$
    if x is consistent with e then
      N$[x] \leftarrow$ N$[x] + 1$ where x is the value of X on x
return Normalize $N[X]$

Likelihood Weighting. This method relies on fixing evidence variables, sampling only non-evidence variables, and weighting each sample by the likelihood according to evidence.

function WeightedSample (bn, e) **returns** an event and a weight
  x $\leftarrow$ an event with n elements; $w \leftarrow 1$
  for $j = 1$ to n do
    if X_i has a value x_i in e
    then $w \leftarrow w \times P(X_i = x_i|Parents(X_i))$
    else $x_i \leftarrow$ a random sample from $P(X_i = x_i|Parents(X_i))$
return x, w

function LikelihoodWeighting (X, e, bn, N) **returns** an approximation to $P(X|e)$
  W[X] $\leftarrow$ a vector of counts over X, initially zero.
  for $j = 1$ to N do
  $x, w \leftarrow WeightedSample(bn)$
    W$[x] \leftarrow$ W$[x] + w$ where x is the value of X on x
return Normalize $W[X]$

2.2 The Independent Choice Logic.

The Independent Choice Logic is used for modeling multiple agents acting under conditions of uncertainty. **ICL** was proposed and developed by David Poole [7]. **ICL** is inspired by game theory, Bayesian networks, probabilistic Horn abduction, Markov decision processes, agent modeling and dynamical systems. In **ICL**, knowledge representation is provided by a symbolic modeling language

which guides the user how to model the domain. In order to use an **ICL** framework for fault diagnosis, it is necessary to represent the power network with a logical model.

System representation in the ICL framework. To illustrate the process of representing the power transmission network in the **ICL** framework, consider the simplified power network shown in figure 2.

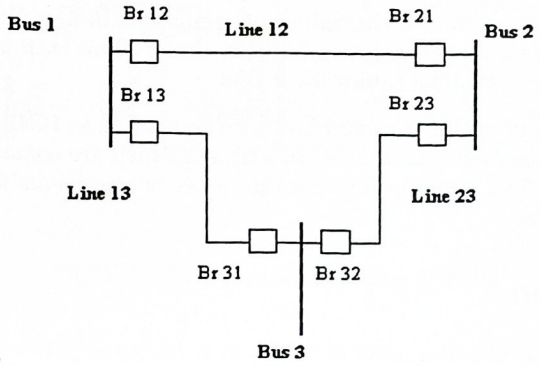


Fig. 2. Single line diagram of a small power network

The first step is constructing the BN to model the dependency between network elements. The random variables without parents are the nodes or buses, and for simplicity, they are assumed to be the only source of faults. The breakers have two parents (buses), because they are the main protection for one bus and the backup protection for another one - for instance breaker Br12, is the main protection for bus1 and the backup protection for bus2 (see figure 1). This scheme of backup protection allows the isolation of a bus fault, even in the event of a main breaker malfunction. From the BN, the following translation procedure is used to model the power network in the **ICL** framework. The procedure is similar to the one developed in [13].

1. The random variables with no parents are encoded as random choices. For instance, *Bus 1* is represented as:

   random([bus1(ok):0.9,bus1(faulted):0.1])

   The last representation states that Bus1 has a 0.9 probability to be in normal state and 0.1 probability to be in a faulted state. Prior probabilities may be

estimated by a human expert or extracted from complete reliability methods.

2. For each random variable $Br_i(V)$ with n parents, there is a rule of the form:

$$Br_i(V) \leftarrow bus_1(V_1) \wedge \cdots \wedge bus_n(V_n) \wedge c_B r_i(V, V_1, \ldots, V_n)$$

The intended interpretation of $c_B r_i(V, V_1, \ldots, V_n)$ is that Br_i has a value V because bus_1 has value V_1, ..., and bus_n has value V_n. For instance, the rule for Breaker *Br12* is:

$$Br12(StBr) \leftarrow bus1(StB1) \wedge bus2(StB2) \wedge c_B r_{12}(StBr, StB1, StB2)$$

3. For each combination of the values of arguments in $c_B r_i(V, V_1, \ldots, V_n)$ variables, there is an assertion as a random choice. This step is similar to filling conditional probability tables for a BN.

Once the power network BN model has been translated to **ICL**, a theorem prover is applied to generate explanations (diagnoses) which are consistent with the set of observations. The explanations contain a set of suspicious faulty elements in the power network.

3 Case Study

The case study is the diagnosis of faults in a 12-buses power transmission network. This is a portion of the network described in [3]. The single line diagram of the tested transmission network is shown in figure 3. The system consist of 12 buses, 16 lines and 32 breakers. The BN representation for the power network is shown on figure 4.

We implement two different methods for inference in **BNs**, to test the performance of each one. In the first implementation we use a probabilistic logic representation (**ICL**) with inference computed by a theorem prover and in the second we use an implementation of the BN on MatLab with inference computed by approximate methods. Single and multiple faults were considered with clean evidence (complete information given by the correct status of protection breakers) and noisy evidence (missed information of some breakers and fail-to-open status in main breakers). Every fault scenario was repeated 10 times for both implementations and average time was recorded.When using **ICL** representation, most likely explanation was used, and in all cases the correct diagnosis was found. In the other case, with approximate inference algorithms, time of diagnosis was recorded when correct diagnosis probability exceeded a 0.5 value threshold. To run the experiments we use a 800 MHz AMD Turion 64 computer with Windows XP operating system and 384 MB of RAM. The approximate inference algorithms were implemented in MatLab v7.2, and the **ICL** code was taken from [8].

The results of experiments show that approximate method by Likelihood Weighting sampling outperform in most cases the other two inference algorithms (see tables 2, 3, 4, and 5). In the other hand, Rejection Sampling performance

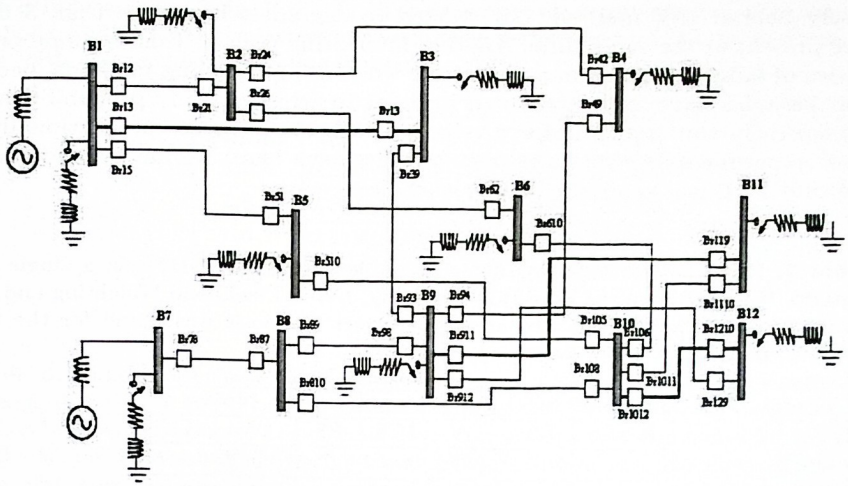


Fig. 3. Power Network single line diagram

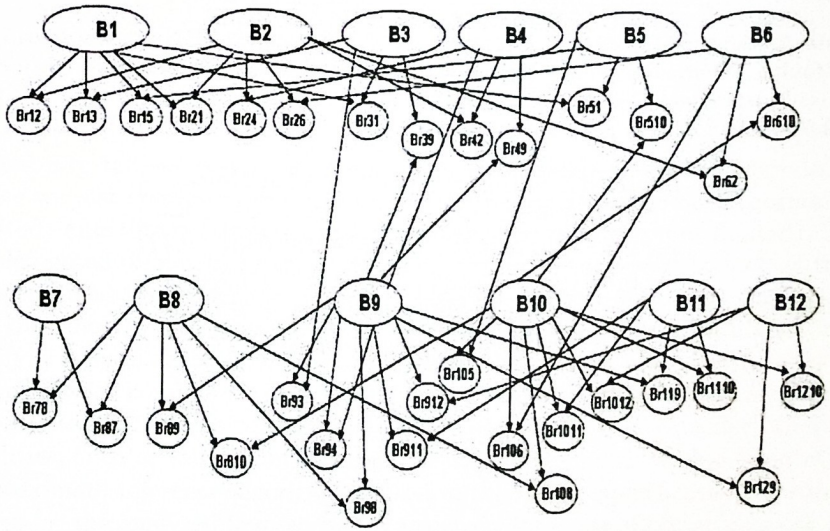


Fig. 4. BN model of the power network

seems competitive when using clean evidence, but is not adequate when noisy evidence is given, as it shows poor results. **ICL** approach performance followed closely behind **LW** method, but started to degrade when more than 3 faults were present at the same time. Another interesting point is that extreme probabilities of failure plays an important role when using sampling methods, because more samples have to be generated in order to achieve steady probabilities and consequently running time for a reliable diagnosis increases proportionally. In other experiments we perform (results not shown here) we have found that in this case **ICL** is a good alternative of choice.

Table 2. Runtime (seconds) for different implementations of **BNs** in a single fault scenario. Where **RS** means Rejection Sampling, **LW** is Likelihood Weighting and **ICL** is Independent Choice Logic. In all cases correct diagnosis was found for the three methods, and **LW** was the fastest approach

Faulted	without Noise			with Noise		
Node	RS	LW	ICL	RS	LW	ICL
2	0.047	0.013	0.305	0.039	0.002	0.266
6	0.052	0.006	0.174	0.034	0.006	0.141
9	0.048	0.006	0.79	0.063	0.005	0.77
11	0.041	0.013	0.17	0.075	0.002	0.142

Table 3. Runtime (seconds) for different implementations of **BNs** in a scenario with two faults. Where **RS** means Rejection Sampling, **LW** is Likelihood Weighting and **ICL** is Independent Choice Logic. In all cases correct diagnosis was found for the three methods, and **LW** was the fastest approach

Faulted	without Noise			with Noise		
Nodes	RS	LW	ICL	RS	LW	ICL
2,3	0.175	0.025	0.638	2.589	0.025	0.47
8,9	0.036	0.023	2.167	0.042	0.030	1.586
3,7	0.039	0.025	0.458	0.036	0.025	0.287
1,10	0.030	0.017	3.928	0.3.35	0.023	3.344

4 Related Work

The closest works to ours are stated in [11,12]. In [11] Bayesian networks with Noisy-or and Noisy-and are used to avoid the specification of big conditional

Table 4. Runtime (seconds) for different implementations of BNs in a scenario with three faults. Where **RS** means Rejection Sampling, **LW** is Likelihood Weighting and **ICL** is Independent Choice Logic. In all cases correct diagnosis was found for the three methods, and **LW** was the fastest approach. The performance of **RS** method was very poor with noisy evidence and was omitted

Faulted	without Noise			with Noise	
Nodes	RS	LW	ICL	LW	ICL
2,3,4	*0.223*	*0.020*	*0.575*	*0.234*	*0.322*
7,8,9	*0.364*	*0.255*	*3.233*	*0.227*	*0.892*
1,3,7	*0.519*	*0.152*	*1.049*	*0.275*	*0.514*
5,8,12	*0.275*	*0.152*	*1.00*	*0.230*	*0.600*

Table 5. Runtime (seconds) for different implementations of BNs in a scenario with three faults. Where **RS** means Rejection Sampling, **LW** is Likelihood Weighting and **ICL** is Independent Choice Logic. In all cases correct diagnosis was found for the three methods, and **LW** was the fastest approach. The performance of **RS** method was very poor with noisy evidence and was omitted

Faulted	without Noise			with Noise	
Nodes	RS	LW	ICL	LW	ICL
1,2,3,4	*0.670*	*0.995*	*1.106*	*1.320*	*0.409*
7,8,9,10	*1.378*	*4.945*	*8.430*	*0.227*	*2.666*
2,6,8,10	*0.970*	*0.600*	*16.028*	*0.770*	*4.902*
3,7,9,12	*0.530*	*0.509*	*3.603*	*0.881*	*1.006*

probability tables. In our case, conditional probability tables are bounded because we just consider two possible states in power network nodes (normal and faulted) and three states on breaker devices (normal, open and faulted). They implemented different models for elements of the power network, such as transformers, busbars and transmission lines, whereas we just consider a single model for the whole network. Another difference is that they apply an algorithm to verify and learn parameters (probabilities) in the models, while in this work this feature is not considered. In [12] a BN model similar to the one presented here is proposed but in this case the main difference is that they consider other possibilities, such as failures on the transmission lines. In this work a MonteCarlo approximate inference algorithm is used, although no specification of running times for the analyzed fault scenarios are given. The size of power network used in their experiments is similar to our network.

Although in both papers realistic cases were tested, no further analysis is made about the quality of evidence or the performance when multiple faults are present on the system.

5 Conclusions

A Power networks fault diagnosis approach, based on Bayesian networks has been presented. We have tested the method on a power network with realistic proportions. We have also implemented two different inference methods which offer different strengths and weaknesses under different circumstances. Rejection Sampling showed a good performance when clean evidence was present, whereas likelihood weighting was the best method with and without noisy evidence. In the other hand, **ICL** method based on a theorem prover seems to be more appropriate when extreme probabilities of failure are used and approximate methods increase in a significant way their running times.

References

1. T. Bi, Y. Ni, and F. Wu: Distributed Fault Section Estimation System for Large-Scale Power Networks. Power Engineering Society Winter Meeting 2002. pp. 1350-1353, vol. 2.
2. L. Garza (2001) : Hybrid Systems Fault Diagnosis with a Probabilistic Logic Reasoning Framework. Instituto Tecnolgico y de Estudios Superiores de Monterrey. PhD thesis 2001.
3. Reliability Test System Task Force, Application of Probability Methods Subcomitee. IEEE Reliability Test System. IEEE Transactions on Power Apparatus and Systems, Vol. 98, No. 6, pp. 2047-2054, 1979.
4. Malheiro, N., Vale Z., Ramos C., Marques A., and Couto V.: On-line Fault Diagnosis with Incomplete Information in a Power Transmisin Network. Proceedings of the 13th International Conference on Intelligent Systems Application to Power Systems 2005. pp. 169-174.
5. McArthur S., and Davidson E.: Automated Post-fault Diagnosis of Power System Disturbances. In Proceedings of the IEEE Power Engineering Society Meeting 2006.
6. Microtran Power Systems Analysis Corporation. Microtran Reference Manual, Vancouver, BC Canada 1997.
7. Poole D.: The Independent Choice Logic for Modeling Multiple Agents under Uncertainty. Artificial Intelligence, Vol. 94, No. 1-2, special issue on economic principles of multi-agent systems, pp. 7-56, 1997.
8. Poole D.: Independent Choice Logic Interpreter version 0.2.1 PROLOG CODE. Technical Report, Dept. of Computer Science, University of British Columbia, 1998.
9. H. Ren, Z. Mi: Power System Fault Diagnosis Modeling Techniques based on Encoded Petri Nets. Power Engineering Society general Meeting 2006.
10. Xu L., and Chow M.: Power Distribution Systems Fault Case Identification using Logistic regression and Artificial Neural Network. Proceedings of the 13th International Conference on Intelligent Systems Application to Power Systems 2005. pp. 163-168.
11. Z. Yongli, H. Limin, and L. Jinling: Bayesian Networks-based Approach for Power systems Fault Diagnosis. IEEE Transactions on Power Delivery, Vol. 21, No. 2, April 2006.
12. W. Zhao, X. Bai, J. Ding, Z. Fang, Z. Li, and Z. Zhou: A New Uncertain Fault Diagnosis Approach of Power System Based on Markov Chain Monte Carlo Method. International Conference on Power System Technology 2006.

An Indirect Adaptive Neural Control of a Wastewater Treatment Bioprocess via Marquardt Learning

Carlos R. Mariaca-Gaspar and Ieroham S. Baruch

Dept. of Automatic Control, CINVESTAV-IPN, Ave. IPN No 2508, A.P. 14-470,
07360 Mexico D.F., MEXICO
{cmariaca, baruch}@ctrl.cinvestav.mx

(Paper received on July 10, 2007, accepted on September 1, 2007)

Abstract. The paper propose a new Recurrent Neural Network (RNN) model for systems identification and states estimation of a highly nonlinear wastewater treatment bioprocess using the recursive Levenberg-Marquardt learning algorithm. The estimated states of the RNN model are used for an indirect adaptive trajectory tracking sliding mode control. The applicability of the proposed control scheme is applied for continuous wastewater treatment bioprocess model, taken from the literature, where a good convergence and a low Mean Squared Error of reference tracking is achieved.

1 Introduction

The rapid growth of available computational resources led to the development of a wide number of Neural Networks (NN)-based modelling, identification, prediction and control applications, [1], [2]. The main network property namely the ability to approximate complex non-linear relationships without prior knowledge of the model structure makes them a very attractive alternative to the classical modelling and control techniques. The neural networks and the neuro-fuzzy based techniques were successfully applied in several engineering areas as: direct model reference adaptive control of MIMO nonlinear processes, [3]; modeling and control of wastewater treatment process, [4]; neuro-fuzzy control of robotic exoskeleton, [5]. The proposed in the literature neural control gives a good approximation of the nonlinear plants dynamics, better with respect to the other methods of control, but the applied static NNs have a great complexity, and the plant order has to be known. The application of Recurrent NNs (RNN) could avoid these problems and could reduce significantly the size of the applied NNs.

In some early papers, [6], [7] the state-space approach is applied to design a RNN in a universal way, defining a Jordan canonical two or three layer RNN model, named Recurrent Trainable Neural Network (RTNN) and the Backpropagation (BP) algorithm of its learning. Then this general RTNN approach is extended using the Levenberg-Marquardt (L-M) algorithm of learning, [8], [9], applied for direct neural control of mechanical and biotechnological plants. In the present paper we go ahead applying an indirect adaptive sliding mode control of the same biotechnological plant, using RTNN identifier learned by the L-M learning algorithm, executed in real-time, [9].

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 203-213

2 RTNN Topology and Levenberg-Marquardt learning algorithm

A Recurrent Trainable Neural Network model and the dynamic BP learning algorithm, together with the explanatory figures and stability proofs, are given in [7]. The RTNN topology (see Fig. 1), given in vector-matrix form is described by the following equations:

$$X(k+1) = JX(k) + BU(k); \quad J = block - diag(J_i); |J_i| < 1 \tag{1}$$

$$Z(k) = \varphi[X(k)] \tag{2}$$

$$Y(k) = \varphi[CZ(k)] \tag{3}$$

where: Y, X, and U are, respectively, output, state and input vectors with dimensions l, n, m; J is a (nxn)- state block-diagonal weight matrix; J_i is an i-th diagonal block of J with (1x1) dimension. The inequality in equation (1) represents the local stability conditions, [7], [8], imposed on all blocks of J; B and C are (nxm) and (lxn)- input and output weight matrices; $\varphi[.]$ is vector-valued sigmoid or hyperbolic tangent-activation function; k is a discrete-time variable. The stability of the RTNN model is assured by the activation functions and by the local stability condition (1).

The recursive L-M algorithm [8]-[12] is derived by incorporating a regularization term to the recursive prediction error algorithm which becomes:

$$R(k) = \alpha(k)R(k-1) + (1-\alpha(k))(\nabla Y[W(k)]\nabla Y^T[W(k)] + \rho I_{N_w}) \tag{4}$$

Unfortunately, the inversion matrix lemma is now no longer practical and one partial but effective solution is to add a small constant ρ to one of the diagonal elements of $\nabla Y[W(k)]\nabla Y^T[W(k)]$ at time as proposed in [11], [12]. The equation (4) can then be expressed as:

$$R(k) = \alpha(k)R(k-1) + (1-\alpha(k))(\nabla Y[W(k)]\nabla Y^T[w(k)] + \rho Z_{N_w}) \tag{5}$$

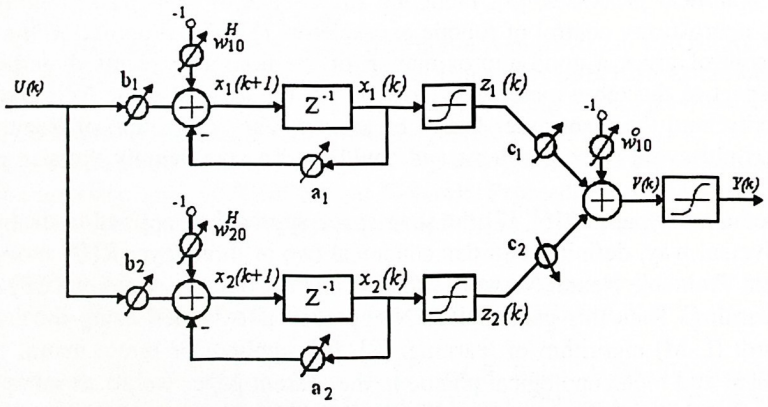


Fig. 1. Recurrent neural network topology (1,2,1).

where Z_{N_w} is a $N_w \times N_w$ diagonal matrix with one non-zero diagonal element which changed from iteration to iteration as follows:

$$z_{ii} = 1, \text{ when}: i = k \bmod(N_w) + 1, \text{ and}: k > N_w \tag{6}$$

$$z_{ii} = 0, \text{ otherwise} \tag{7}$$

With this modification the expression (4) can be re-written in a concise form as:

$$R(k) = \alpha(k)R(k-1) + (1-\alpha(k))(\Omega[W(k)]\Lambda^{-1}(k)\Omega^T[W(k)]) \tag{8}$$

where $\Omega[w(k)]$ is a $N_w \times 2$ matrix with the first column corresponding to $\nabla Y[W(k)]$ and the second column consist of a $N_w \times 1$ vector with one element set to 1, in accordance with equations (9) and (10) above, as it is:

$$\Omega^T[W(k)] = \begin{pmatrix} \nabla Y^T[W(k)] \\ 0 \quad .... \quad 1 \quad .... \quad 0 \end{pmatrix}, \text{ and } \Lambda^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & \rho \end{pmatrix} \tag{9}$$

The matrix inversion lemma can now be applied to (8) leading to following recursive Levenberg-Marquardt formulation:

$$S[W(k)] = \alpha(k)\Lambda(k) + \Omega^T[W(k)]P(k-1)\Omega[W(k)] \tag{10}$$

$$P(k) = \frac{1}{\alpha(k)}[P(k-1) - P(k-1)\Omega[W(k)]S^{-1}[W(k)]\Omega^T[W(k)]P(k-1)] \tag{11}$$

Typically the choice of α is in the limits: $0.95 < \alpha < 1$.

As the recursive L-M is based on the Newton method of optimization, it does not needs a stability proof. Next the given up topology and learning are applied for identification and sliding mode control of wastewater treatment bioprocess.

3 Indirect Adaptive Sliding Mode Control Systems Design

Based on the state and parameter estimations, performed by the RTNN identifier, we could propose the following indirect adaptive control scheme, depicted in Fig. 2. The block diagram of that control contained a RTNN identifier and a linear Sliding Mode Controller, designed using the estimated weight parameters.

Let us suppose that the studied nonlinear plant possess the following structure:

$$X_p(k+1) = F(X_p(k), U(k)) \tag{12}$$

$$Y_p(k) = G(X_p(k)) \tag{13}$$

where: $X_p(k)$, $Y_p(k)$, $U(k)$ are plant state, output and input vector variables with dimensions Np, L and M, where L=M is supposed; F and G are smooth, odd, bounded

nonlinear functions. The linearization of the activation functions of the learned identification RTNN model, which approximates the plant (see equations (1) to (3)), leads to the following linear local plant model:

$$X(k+1) = JX(k) + BU(k) \tag{14}$$

$$Y(k) = CX(k) \tag{15}$$

where L=M, is supposed.

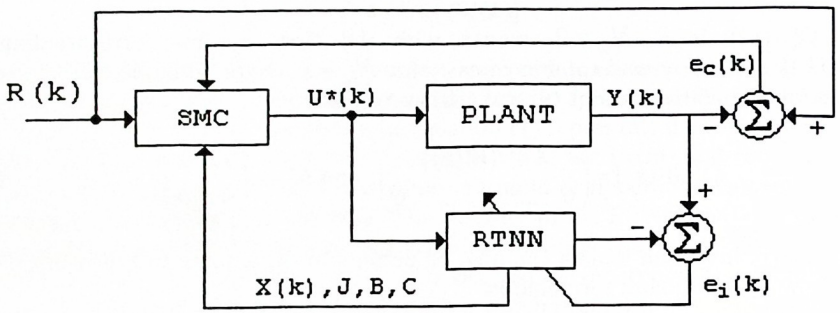


Fig. 2. Block - diagram of the closed-loop system containing neural identifier and sliding mode controller.

Let us define the following sliding surface with respect to the output tracking error:

$$S(k+1) = E(k+1) + \sum_{i=1}^{P} \gamma_i E(k-i+1) \; ; \; |\gamma_i| < 1 \tag{16}$$

where: S(.) is the sliding surface error function; E(.) is the systems output tracking error; γ_i are parameters of the desired error function; P is the order of the error function. The additional inequality in (16) is a stability condition, required for the sliding surface error function. The tracking error in two consecutive steps is defined as:

$$E(k) = R(k) - Y(k) \; ; \; E(k+1) = R(k+1) - Y(k+1) \tag{17}$$

where R(k) is an L-dimensional reference vector and Y(k) is an output vector with the same dimension.

The objective of the sliding mode control systems design is to find a control action which maintains the systems error on the sliding surface which assure that the output tracking error reaches zero in P steps, where P<N. The control objective is fulfilled if:

$$S(k+1) = 0 \tag{18}$$

Now, let us to iterate (15) and to substitute (14) in it so to obtain the input/output local plant model, which yields:

$$Y(k+1) = CX(k+1) = C[JX(k) + BU(k)] \tag{19}$$

From (16), (17), and (18), it is easy to obtain:

$$R(k+1)-Y(k+1)+\sum_{i=1}^{P}\gamma_i E(k-i+1)=0 \tag{20}$$

The substitution of (19) in (20) gives:

$$R(k+1)-CJX(k)-CBU(k)+\sum_{i=1}^{P}\gamma_i E(k-i+1)=0 \tag{21}$$

As the local approximation plant model (14), (15), is controllable, observable and stable, see [7], the matrix J is diagonal, and L=M, the matrix product (CB) is nonsingular, and the plant states X(k) are smooth non- increasing functions. Now, from (21) it is easy to obtain the equivalent control capable to lead the system to the sliding surface which yields:

$$U_{eq}(k)=(CB)^{-1}\left[-CJX(k)+R(k+1)+\sum_{i=1}^{P}\gamma_i E(k-i+1)\right] \tag{22}$$

Following [13], the SMC avoiding chattering is taken using a saturation function inside a bounded control level Uo, taking into account plant uncertainties. So the SMC takes the form:

$$U(k)=\begin{cases} U_{eq}(k), & if\ \|U_{eq}(k)\|<U_0 \\ -U_0 U_{eq}(k)/\|U_{eq}(k)\|, & if\ \|U_{eq}(k)\|\geq U_0 \end{cases} \tag{23}$$

The proposed SMC cope with the characteristics of the wide class of plant model reduction neural control with reference model, defined by Narendra, [1], and represents an indirect adaptive neural control, given by Baruch, [7].

4 Description of the biological Wastewater Treatment Bioprocess

Wastewater treatment is performed in an aeration tank, in which the contaminated water is mixed with biomass in suspension (activated sludge), and the biodegradation process is then triggered in the presence of oxygen. The tank is equipped with a surface aeration turbine, which supplies oxygen to the biomass, and additionally changes its suspension into a homogeneous mass. After some period, the biomass mixture and the remaining substrate go to a separating chamber where the biologic flocks (biologic sludge) are separated from the treated effluent. The treated effluent is then led to a host environment. The aim is good settling of the biomass in the settler and high conversion of the entering organic material in the bioreactor (see Fig. 3). The main objective of the control system is to keep the recycle biomass concentration close to the reference signal, and this should be achieved in the presence of disturbances and measurement noise

acting on the recycle flow rate. A detailed description of all reactions arising in the bioreactor would lead to a high-order model of differential equations [14]. For the control strategy developed in this work a simplified reduced order model is sufficient, as far as it preserves the structural properties of the process, [15]. The model equations are derived using the mass balance of the bioreactor and the settler.

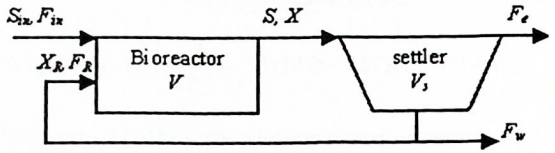


Fig. 3. Biological wastewater treatment with settler.

Mass balance to the bioreactor. The obtained equations are as follows:

$$\dot{X}(t) = \left(\mu(S) - \frac{F_{in}(t) + F_R(t)}{V} - c_d(t) \right) X(t) + \frac{F_R(t)}{V} X_R(t) \tag{24}$$

$$\dot{S}(t) = -\frac{1}{Y} \mu(S) X(t) + \frac{F_{in}(t)}{V} S_{in} - \frac{F_{in}(t) + F_R(t)}{V} S(t), \tag{25}$$

where the state variables are: X(t), biomass concentration; S(t), the substrate measured by the Chemical Oxygen Demand (COD); V is the reactor volume; F_R represents the recycle flow rate (manipulated variable), F_{in} is the influent flow rate; S_{in} is the influent substrate concentration (potential disturbance, also expressed as COD), and Y>0 is the yield coefficient. Here c_d X denotes the decay rate of the biomass concentration (which is added in the model to simulate biomass mortality), with $c_d > 0$ as the decay rate parameter. The variable $\mu(\cdot)$ (specific growth rate) is modeled by a Mono-type equation:

$$\mu(S(t)) = \frac{\mu_m(t) S(t)}{K_m(t) + S(t)} \tag{26}$$

where: $\mu_m(\cdot)$ is the maximum growth rate and $K_m(\cdot)$ is the half-saturation constant of biodegradable organic matter. It is the concentration of the substrate for which $\mu = \mu_m/2$. Both parameters are subject to variations.

Mass balance to the settler. It is supposed that none of the biomass is left in the effluent F_e of the settler (see the Fig. 3), so that the whole biomass in the clarifier is settled. The dynamics of the concentration of the biomass in the settler, X_R (t), can be described by the following mass balance equation:

$$\dot{X}_R(t) = \left(\frac{F_{in}(t) + F_R(t)}{V_S} \right) X(t) + \left(\frac{F_W(t) + F_R(t)}{V_S} \right) X_R(t) \tag{27}$$

where: F_W denotes the waste flow rate and V_s is the volume of the settler. We can approximate the settler behavior by:

$$X_R(t) = q(t)X(t) \tag{28}$$

where the parameter q(t) is considered as continuously differentiable and bounded function with bounded inverse, bounded derivative, and q(t) >1 for all t ≥ 0.

Process measurements. The sensor dynamics is modeled by:

$$T_{mm}\dot{X}(t) = -X_m(t) + X_R(t) + n(t) \tag{29}$$

The bioprocess dynamics is corrupted by some white Gaussian noise n(t). The specific model, we consider for the simulations, is obtained after substitution of $X_R(t)$ from equation (28) into equation (24) and has the following form:

$$\dot{X}_R(t) = \left(\frac{\dot{q}(t)}{q(t)} + \mu(t, S(t)) - \frac{F_{in}(t)}{V} - c_d + \frac{q(t)-1}{V}\right) X_R(t) \tag{30}$$

$$\dot{S}(t) = -\frac{1}{Y(t)}\mu(t, S(t))\frac{1}{q(t)}X_R(t) + \frac{F_{in}(t)}{V}S_{in} - \frac{F_{in}(t)+F_R(t)}{V}S(t) \tag{31}$$

Time-varying control reference. The control objective is to assure that the biomass concentration in the recycle flow tracks asymptotically a time-varying reference signal, which is proportional to the influent flow rate and it is assumed to be measurable:

$$X_{R\,ref}(t) = k_{ref}F_{in}(t) \tag{32}$$

The specific model considered for process simulation is the system of nonlinear differential equations (29), (30), (31), and the Monod-type equation (26), with constant parameters:

$$V = 1{,}5.107\ [\mathrm{l}],\ S_{in} = 300\ [\mathrm{mg\ COD\ /l}],\ T_m = 1/12\ [\mathrm{h}] \tag{33}$$

The model uncertainties are taken into account by introducing time-varying parameters as:

$$\mu_m(t) = 0.2 + 0.1\sin(2\pi t/3 + 4\pi/3);\ K_m(t) = 90 + 30\sin(\pi t/2) \tag{34}$$

$$Y(t) = 0.6 + 0.1\sin(\pi t/3 + \pi/3);\ q(t) = 4 + \sin(\pi t/6); \tag{35}$$

$$c_d(t) = 10^{-4}(25 + 5\cdot\sin(\pi t/12)) \tag{36}$$

The control objective is to track the reference signal, given by the equation (32), where the parameters are as follows:

$$k_{ref} = 3.8*10^{-3}\ [\mathrm{mgh/l2}];\ F_{in}(t) = 3*10^6\left(1 + 0.25\sin\pi t/12\right) \tag{37}$$

The initial conditions are always set to:

$$S(0) = 8 \text{ (mgCOD/l)}, \quad X_R(0) = 11.4 * 10^3 \text{ (mg/l)}, \quad X_m(0) = 0 \text{ (mg/l)} \tag{38}$$

In order to overcome saturation of the RTNNs, the output and the input of the plant are scaled by the following procedure:

$$y_p = (X_m - 11400)/5700 \, ; F_R = \left[\left(\left[\left(U * 7.5 \times 10^5 \right) + 3 \times 10^6 \right] F_{Conv} \right) - X_m \right] K_{stab} \tag{39}$$

where the scaling parameters are given by: K_{stab}= 3*10-3 , F_{conv}= 0.0038. These scale factors correspond to the range of the reference signal. Note that the variable U is the bioreactor input control signal, generated by one of the proposed control algorithms, while F_R is the physical process input (the recycle flow rate). Analogous, y_p is the scaled output of the bioreactor, while X_m is the real measured output. Consequently, the reference signal is also normalized following the same procedure (see equations (32) and (37). The reference signal is given by:

$$r(k) = \left(X_{Rref}(k) - 11400 \right) / 5700 \tag{40}$$

Hence, substituting (37), (44) into (32), and the obtained result in (39), the scaled reference signal is obtained as:

$$r(k) = 0.5 \sin\left(\pi k / 12 \right) \tag{41}$$

The inverse transformation of (46) and the physical bioprocess control signal are as:

$$X_m = 5700 y_p + 11400 \, ; F_R = \left(0.5U - y_p \right) 1.71 \times 10^8 \tag{42}$$

Note that the recycle flow rate F_R is a function of the control variable U, computed by the feedback control with respect to the estimated state and the feedforward control with respect to the scaled plant reference r(k).

5 Simulation Results

A graphical simulation results obtained with the described above wastewater treatment biotechnological plant, obtained using the given up identification and sliding mode control methodology, are shown on Fig. 4. All simulations are performed using the following set of equations: the process description (29), (30), (31); the Mono-type equation (26); the time variable plant parameters (34)–(36); the plant output scaling equation (39); the scaled reference signal equation (41); the scaled plant input equation (42). In all simulations, the severe realistic conditions such as measurement noise are

taken into account by generating a stochastic signal added to the process input and output, both with variance 1200, which is commonly used to simulate noisy measurements.

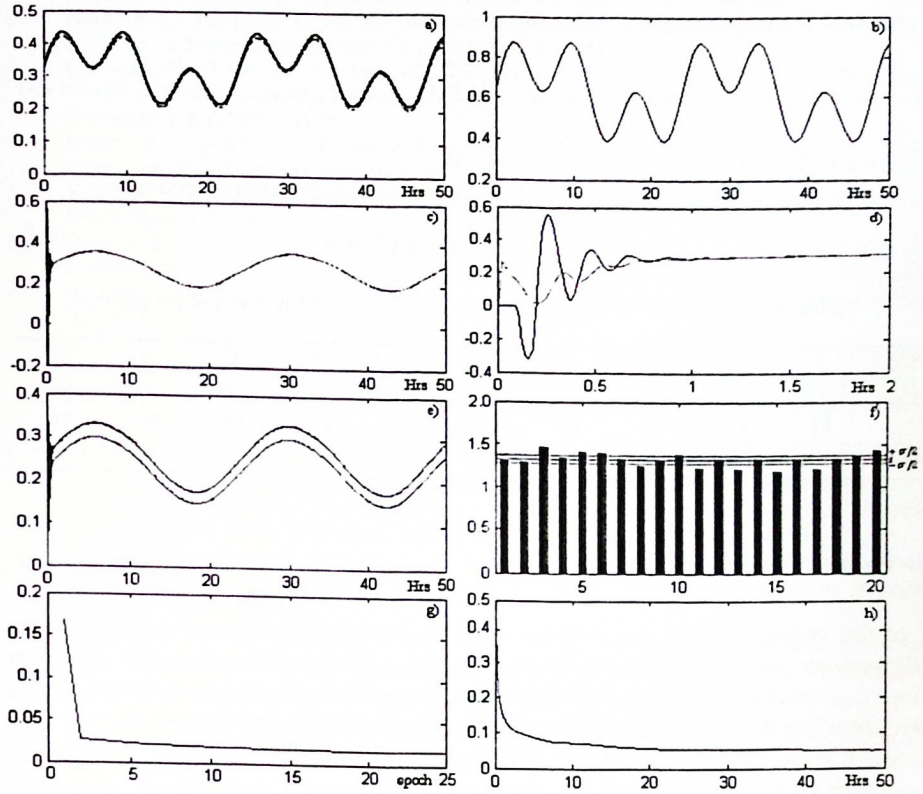


Fig. 4. Graphical results obtained using Indirect SMC and L-M algorithm a) Comparison between the plant output and the reference signal of the control system; b) The control signal; c) System Identification; d) Same as c), but during the first 2 hours. e) States of the plant; f) MSE of control obtained during 20 simulations; g) MSE of identification during 25 epochs; h) The MSE of indirect sliding mode control.

The variance chosen corresponds to 10% noise on the data. The process is simulated over a period of 50 hours, which gives an idea about its periodic behavior (a typical period is about 24 hours) and the period of discretization is set to T_0=0.01h (it is 1 hour of the process time). The parameter used is α=0.95. The activation functions of the hidden and output network layers are hyperbolic tangents. The identification RTNN has topology (1, 2, 1). The results show a good convergence of the system output to the desired trajectory after approximately 1.5 h and a good filtration of the noise which makes a MSE% reduction up to 0.5%. The behavior of the control system

in the presence of 10% white Gaussian noise on the plant output could be studied accumulating some statistics of the final MSE% (ξ_{av}) for multiple run of the control program which results are given on Table 1 for 20 runs. The mean average cost for all runs (ε) of control, the standard deviation (σ) with respect to the mean value and the deviation (Δ) are given by the following formulas:

$$\varepsilon = \frac{1}{n}\sum_{k=1}^{n} \xi_{av_k}; \quad \sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \Delta_i^2}; \quad \Delta = \xi_{av_k} - \varepsilon \tag{43}$$

Where k is the run number and n is equal to 20.

The mean and standard deviation values of process control are respectively:

$$\varepsilon = 1.3324\ \%; \quad \sigma = 0.0719\ \% \tag{44}$$

Table 1. Final MSE % of control (ξ_{av}) for 20 runs of the control program.

No.	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
MSE	1.322	1.302	1.468	1.342	1.422	1.402	1.342	1.268	1.314	1.386
No.	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
MSE	1.228	1.342	1.222	1.348	1.214	1.341	1.244	13.42	1.382	1.434

6 Conclusions

In this paper a RTNN model and a dynamic Levenberg-Marquardt learning algorithm are proposed to be applied for identification and state estimation of a nonlinear bioprocess plants. The proposed RTNN model has a Jordan canonical structure, which permits to use the generated vector of estimated states directly for process control. The obtained states are used to design an indirect sliding mode control law. It performs very well under restrictive conditions of periodically acting disturbances, parameter uncertainties and inevitable sensor dynamics. The simulation results, obtained with a continuous wastewater treatment bioprocess plant model, taken from the literature, confirm the applicability of the proposed identification and control methodology.

Acknowledgements

The Ph.D. student Carlos-Roman Mariaca-Gaspar is thankful to CONACYT-MEXICO for the scholarship received during his studies in the Department of Automatic Control, CINVESTAV-IPN, Mexico.

References

1. Narendra, K.S., Parthasarathy, K.: Identification and Control of Dynamic Systems Using Neural Networks. IEEE Transactions on Neural Networks, 1(1) (1990) 4-27.
2. Hunt, K.J., Sbarbaro, D., Zbikowski, R., Gawthrop, P.J.: Neural Network for Control Systems - a Survey. Automatica, 28 (6) (1992) 1083-1112.
3. Frayman, Y., Wang, L.P.: A Dynamically-Constructed Fuzzy Neural Controller for Direct Model Reference Adaptive Control of Multi-Input-Multi-Output Nonlinear Processes. Soft Computing, 6 (2002) 244-253.
4. Boger, Z.: Application of Neural Networks to Water and Wastewater Treatment Plant Operation. ISA Trans. 31 (1) (1992) 25-33.
5. Kiguchi, K., Tanaka, T., Fukuda, T.: Neuro-Fuzzy Control of a Robotic Exoskeleton with EMG Signals. IEEE Trans. Fuzzy Systems 12 (2004) 481-490.
6. Baruch, I., Flores, J. M., Thomas, F., Garrido, R.: Adaptive Neural Control of Nonlinear Systems. In: Dorffner, G., Bischof, H., Hornik, K. (eds.): Artificial Neural Networks-ICANN 2001, Lecture Notes in Computer Science 2130, Springer, Berlin, ISBN 3-540-42486-5 (2001) 930-936.
7. Baruch, I., Flores, J. M., Nava, F., Ramirez, I. R., Nenkova, B.: An Advanced Neural Network Topology and Learning, Applied for Identification and Control of a D.C. Motor. In: Sgurev, V., Jotsov, V. (eds.): Proc. of the First International IEEE Symposium on Intelligent Systems, Varna, Bulgaria (2002) 289-295.
8. Baruch, I., Escalante, S., Mariaca, C. R. RNN Identification and Control of Nonlinear Plants Using the Recursive Marquardt Algorithm. In: J.H.Sossa Azuela, R.Barron Fernandez (eds.) Research in Computing Science, Special Issue: Neural Networks and Associative Memories, ISSN: 1870-4069, IPN, CIC, Mexico City, 21 (2006) 151- 160
9. Baruch, I., Escalante, S., Mariaca, C. R., Barrera, J. Recurrent neural control of wastewater treatment bioprocess via Marquardt learning. In: 10th Computer Applications in Biotechnology, 1 (2007) 285-290
10. Asirvadam, V. S., McLoone, S. F., Irwing, G. W.: Parallel and Separable Recursive Levenberg-Marquardt Training Algorithm. In: Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing, (2002) 129-138.
11. Ngia, L. S., Sjöberg J., Viberg, M.: Adaptive Neural Nets Filter Using a Recursive Levenberg-Marquardt Search Direction. IEEE Signals, Systems and Computer, 1 (1998) 697-701.
12. Ngia, L. S., Sjöberg J., Efficient training of neural nets for nonlinear adaptive filtering using a recursive Levenberg-Marquardt algoritm. IEEE Trans. on Signal Processing, 48 (2000) 1915-1927.
13. Young, K.D., Utkin, V.I., and Ozguner, U.: A Control Engineer's Guide to Sliding Mode Control. IEEE Trans. on Control Systems Technology, 7 (3), (1999) 328-342.
14. Sundstrom, D.W., Klei, H.E.: Wastewater Treatment. Prentice-Hall, Englewood Cliffs, NJ (1979).
15. Georgieva, P., Ilchmann, A.: Adaptive λ-Tracking Control of Activated Sludge Processes. Int. J. of Control, 74(12) (2001) 1247-1259.

The Simulation Model of Highpressure, Suspension Waterjet Cutting Process of Marble

Andrzej Perec[1],

[1] Koszalin University of Technology,
Institute of Mechatronics, Nano- and Vacuum Technology,
Raclawicka Street 15-17, 75-620 Koszalin, Poland
andrzej.perec@tu.koszalin.pl
http://www.imnitp.tu.koszalin.pl/

(Paper received on June 25, 2007, accepted on September 1, 2007)

Abstract. In this article, the usage of artificial neural networks for the marble's cutting process simulation by a hydroabrasive suspension waterjet, which pressure has been reduced to 30 MPa, are presented. A triple layer neural network of the perceptron type, which is taught by the errors backward propagation algorithm, has been applied. The article provides detailed description of neural network. This neural network simulates the marble treatment process and predicts its efficiency due to given parameters. Impact of the most important parameters, as pressure, traverse speed, abrasive flow rate, length and diameter of nozzle was shown. The process parameters, which allow to achieve the maximum cut's depth have been determined.

1 Introduction

In the article refers the simulation model cutting of hydroabrasive suspension waterjet cutting process of marble [3]. Laboratory investigations were carried out on test stand [4] has been built from two containers and four independent hydraulic branches, which enable an adjustment of the basic flow parameters. Each branch consists of the following valves: a cut-off valve, a throttle valve, a non-return valve and a manometer. An overflow valve performs the function of an element preventing an excessive increase of pressure. It is set at the pressure of 30 MPa.
A hydraulic monitor P26 type is the source of a high pressure. It is made on the basis of elements of a plunger pump made by an WOMA company. It makes it possible to obtain the maximum pressure of 75 MPa with the rate of water flow of 75 dm^3/min.
The materials were cut by directing the hydroabrasive jet perpendicular to the machined material [6], and then a rectilinear traverse speed in relation to the working nozzle. The thickness of the samples was selected in such a way that, with the most effective machining parameters, cutting through these should not occur, which would make it difficult to correctly determine the depth of the cut.
Rock used for tests, marble, is a metamorphic rock resulting from the metamorphism of limestone composed mostly of calcite (a crystalline form of calcium carbonate, $CaCO_3$). It is extensively used for sculpture, as a building material, and in many other applications. The word 'marble' is colloquially used to refer to many other stones that

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 215-224

are capable of taking a high polish. This metamorphic process causes a complete recrystallization of the original rock into an interlocking mosaic of calcite, aragonite and/or dolomite crystals. The temperatures and pressures necessary to form marble usually destroy any fossils and sedimentary textures present in the original rock.
Pure white marble is the result of metamorphism of very pure limestone. The characteristic swirls and veins of many colored marble varieties are usually due to various mineral impurities such as clay, silt, sand, iron oxides, or chert which were originally present as grains or layers in the limestone.

2 Artificial Neural Network

The artificial neuron is the basic unit of the artificial neuronal net similarly as in the case of neuronal biological nets, nervous cell is the basic unit. The properties of the artificial neuron answer is the most important [5] properties of the biological neuron. You should always remember that artificial equivalents functions are very simplified [8] in the relation to real nervous cells. The artificial neuron makes up the kind of the converter about many entries and one exit. One can distinguish two blocks of the processing of the information inside him. The first is the adding up block, where input signals are multiplied by there importance and added up.
The topology of the net consisting from 5 neurons of the input layer, 30 neurons of hidden layer and one outputs neuron (Fig.1.) was accepted to prediction [1] the water-jet cutting process.

The entrance data directed to the input layer present the most important parameters of the cutting process, which efficiency depends on them. These are:

- **Jet pressure**, at the same time determining its speed and kinetic energy. Because the jet energy grows with the square of the speed, it is the most important parameter. And therefore increasing of the pressure causes the increase of the cut's depth.
- **Abrasive flow rate**, influencing on the jet's kinetic energy. Increasing of the abrasive flow rate causes the increase of energy and cut's depth. But the excessive increasing may cause the drop of stream's energy, as a result of the unfavourable interaction between grains.
- **Traverse speed**, determining the contact time of abrasive grains with the target. Increasing of the traverse speed generally leads to the fact that less abrasive grains are being put in touch with the target, what causes the less efficiency of the cutting process. Reducing of the traverse speed leads to the increasing of the slot's cutting depth.
- **The length of a nozzle** is directly connected with obtaining maximum energy by the jet. In a nozzle, which is too short, the jet is not able to achieve a proper speed (to speed up). However, a too long nozzle may cause the drop of energy, because of the hydraulic losses and rubbing by abrasive during the flow.
- **The nozzle's diameter**, connected with the abrasive flow rate, pressure and hydraulic power. This value is also connected with the width of a cutted slot. The aim is to minimize the diameter, in order to achieve the maximum depth in the range of determined capacity of the cutted slot.

The cut's depth has been used as an output parameter. It is parameter, which describes in the best way abrasive properties of the jet. And therefore it is most often used for this purpose. Optimisation of the cutting process is usually done in order to achieve the maximum cut's depth.

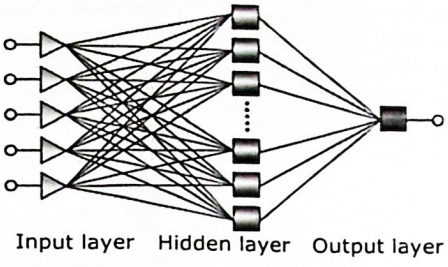


Fig. 1. Artificial Neural Network digram

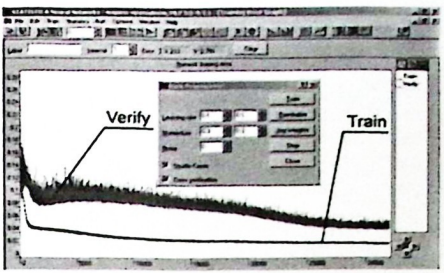


Fig. 2. Diagram of learning the ANN

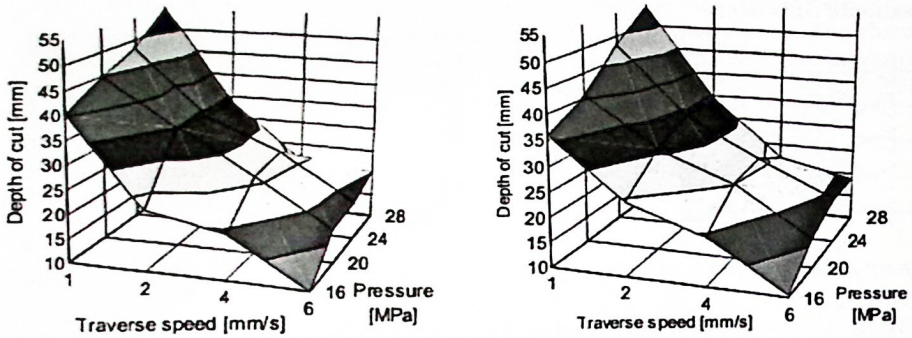
In the hidden layer neuron has logistic activation function. This is an S-shaped (sigmoid) curve, with output in the range (0,1). The most commonly - used neural network activation function. Neurons in input and output layer have linear activation function. The quantity of input and output neurons were taken from the accessible results of investigations directly. 96 laboratory tests results, containing all 5 input values and one output value, have been used for the teaching process of neural network. From process of training excluded 10% of chances, which one used to verification of training process. The net was learning with the algorithm of backward propagation, getting stable results after 30 000 iterations (Fig. 2.) with learning rate of 0.1 and momentum 0.3. To research [2] was utilized the commercial Statistica Neural Networks for Windows application of the StatSoft Inc company.

3 Effects of Artificial Neural Network Modelling

Fig. 3b depicts the results of the artificial neural networks modeling of hydroabrasive suspension jet cut in a variable pressure and traverse speed conditions. In comparison Fig. 3a presents the laboratory analysis in which the surface was adjusted using the least square method. The graphs in the whole show a great convergence in the material cut depth values, a near identical character of dependence and approximate maximum value. In this case maximal between modeled and laboratory values does not go beyond 4.34mm and average discrepancy is equal 0.76mm.

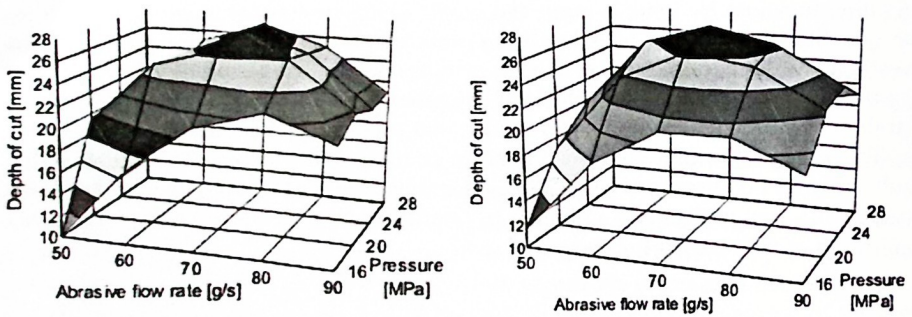
Laboratory studies results, conditioned by variable pressure and abrasive flow rate conditions, are presented in Fig. 4a. Modeling effects are shown in a Fig. 4b. In this case, it can be also observed that modeling effects are compatible with lab studies. The greatest discrepancy is observed at the maximum pressure and abrasive flow rate. In this case standard maximal discrepancy between modeled and laboratory values does not go beyond 4.04mm and average value is 0.72mm. Fig. 5a depicts a laboratory study on marble cutting with the use of 50mm long nozzle while Fig. 5b depicts artificial neural networks modeling of that process. Here also a great modeling and lab

studies compatibility is observed. At maximum cut depth deviation does not exceed 1mm. In this case maximal discrepancy between modeled and laboratory values does not go beyond 1.98mm, and average discrepancy is 0.35mm



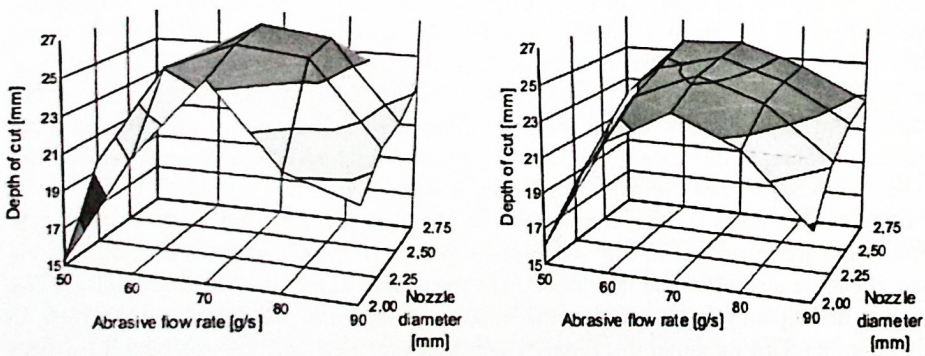
a) laboratory analysis b) modeled with the use of ANN

Fig. 3. Influence traverse speed and pressures onto depth of cutting.



a) laboratory analysis b) modeled with the use of ANN

Fig. 4. Influence abrasive flow rate and pressures onto depth of cutting.



a) laboratory analysis b) modeled with the use of ANN

Fig. 5. Influence abrasive flow rate and 50mm nozzle diameter onto depth of cutting.

Cutting with the use of 75mm long nozzle is presented in Fig. 6a, while its artificial neural network modeling is shown in graph 6b. What can be observed here is, that the modeling, compatible with the laboratory studies (best compatibility at the extreme values). The character of variability is a little different, but the values of maximum cut's depth (the deviation doesn't exceed 2mm) reached at the abrasive flow rate 80g/s and the nozzle diameter about the 2mm, are similar.

With the maximum nozzle diameter, the smallest cut was achieved at the minimum abrasive flow rate. In this case maximal discrepancy between modeled and laboratory values does not go beyond 3.13mm and average discrepancy is 0.27mm

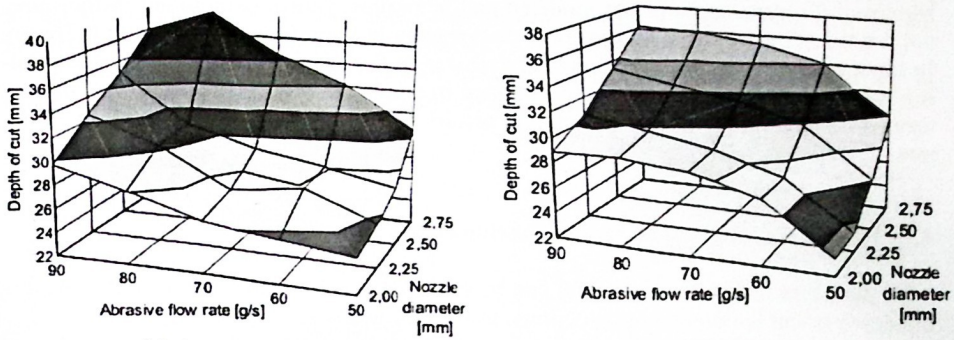


Fig. 6. Influence abrasive flow rate and nozzle 75mm diameter onto depth of cutting.

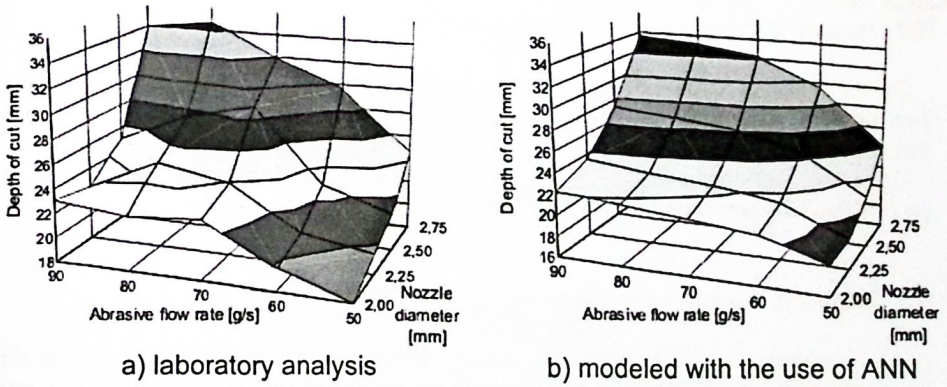


Fig. 7. Influence abrasive flow rate and 100mm nozzle diameter onto depth of cutting.

A laboratory study on cutting with the use of 100mm long nozzle is presented in Fig.7a, while its artificial neural network modeling is presented in Fig. 7b. In this range, modeling effects are also compatible with lab studies. Best compatibility was achieved at the low working nozzle diameters and low abrasive discharge. Main treatment parameter – cut depth – peaks at the maximum abrasive flow rate and minimum

working nozzles diameters. In this case maximal discrepancy between modeled and laboratory values does not go beyond 2.74mm and average value is equal 0.2mm

4 Simulation of the Cutting Process

The artificial neural networks use in the cut depth designating, give similar estimates in every considered case. The divergences do not go beyond 6%. The remaining parameters modeling results do not exceed 5%. In some cases, the discrepancy is at 10%. Maximal discrepancy between modeled and laboratory values is included in the interval from 1.98 to 4.34mm and average discrepancy in the interval from 0.2 to 0,76mm. In most of the cases, the variation character due to the artificial neural network modeling was compatible with the results obtained in empirical way. This will allow application of this model [7] to forecast cutting results for all the variable machining parameters.

4.1 Impact of Traverse Speed on Cutting Depth

On the bases of obtained results it can be stated that in the whole range of parameters the depth of cut is inversely proportional to the cutting speed (Fig. 3.).

Maximum depth of cut was obtained at the minimum traverse speed of 1mm/s and this was accepted as the optimum speed.

4.2 Impact of Pressure on Cutting Depth

On the bases of obtained results it can be stated that in the whole range of tested parameters the depth of cut is either proportional to the working pressure (Fig. 3.) or that the highest pressure will result in the greatest cutting depth (Fig. 4.).

Optimum working pressure will be maximum pressure, in this case equal to p=28MPa. This pressure was used for further simulation work.

4.3 Impact of Abrasive Flow Rate on Cutting Depth

The optimum for this parameter is not as obvious as for the previous two. On the bases of results shown in Fig's 4&5 it could be concluded that the optimum flow rate is ma=70g/s but results obtained and shown in Fig's 6&7 allow only to conclude that increasing the abrasive flow rate results in only marginal increase in cutting depth.

For a 50mm long nozzle the relationship of the abrasive flow rate to the cutting depth is linear and inversely proportional. Increasing the length of the nozzle decreases this relationship leading to flattening of the graph which is greatest for a 100mm long nozzle. On the bases of those results the optimum abrasive flow rate was established to be not less than 70g/s.

4.4 Impact of Nozzle Length and Diameter on Cutting Depth.

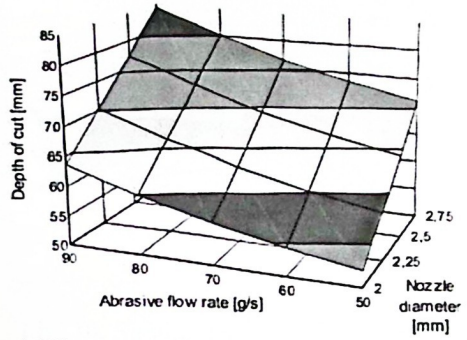
In the case of the shortest length nozzle l=50mm the cutting depth is almost independent of the diameter (Fig. 8a.) in the whole range of abrasive flow.

For l=75mm (Fig. 8b.) we can observe a definite relationship between the depth of cut and diameter of the nozzle for the whole range of the abrasive flow rates.

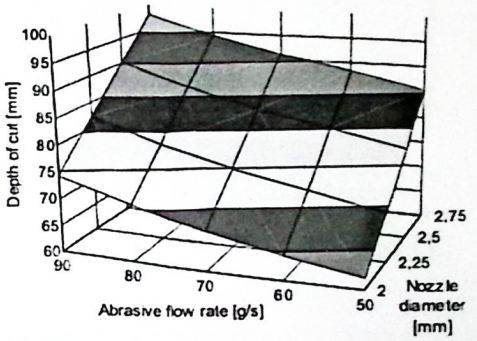
Depth of cut increases with the increase of nozzle diameter. For the longest nozzle l=100mm (Fig. 8c.) this relationship is even stronger.

Observing relationship between the nozzle's length and diameter and depth of cut (Fig. 8, 9 &10.) we observe that the optimum length is 100mm for the whole range of abrasive flow rates. This is least noticeable for low flow rates to the extent that for the lowest values it approximates shorter nozzles.

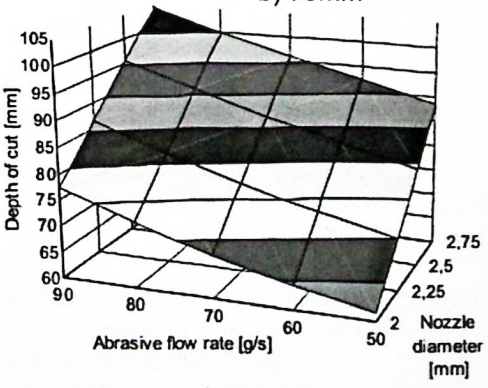
In the whole range of this analyses maximizing the nozzle diameter resulted in cutting depth increase and for this reason the optimum diameter is ϕ=2.75.



a) 50mm



b) 75mm



c) 100mm

Fig. 8. Simulation of influence abrasive flow rate and nozzle diameter about variable length onto depth of cutting.

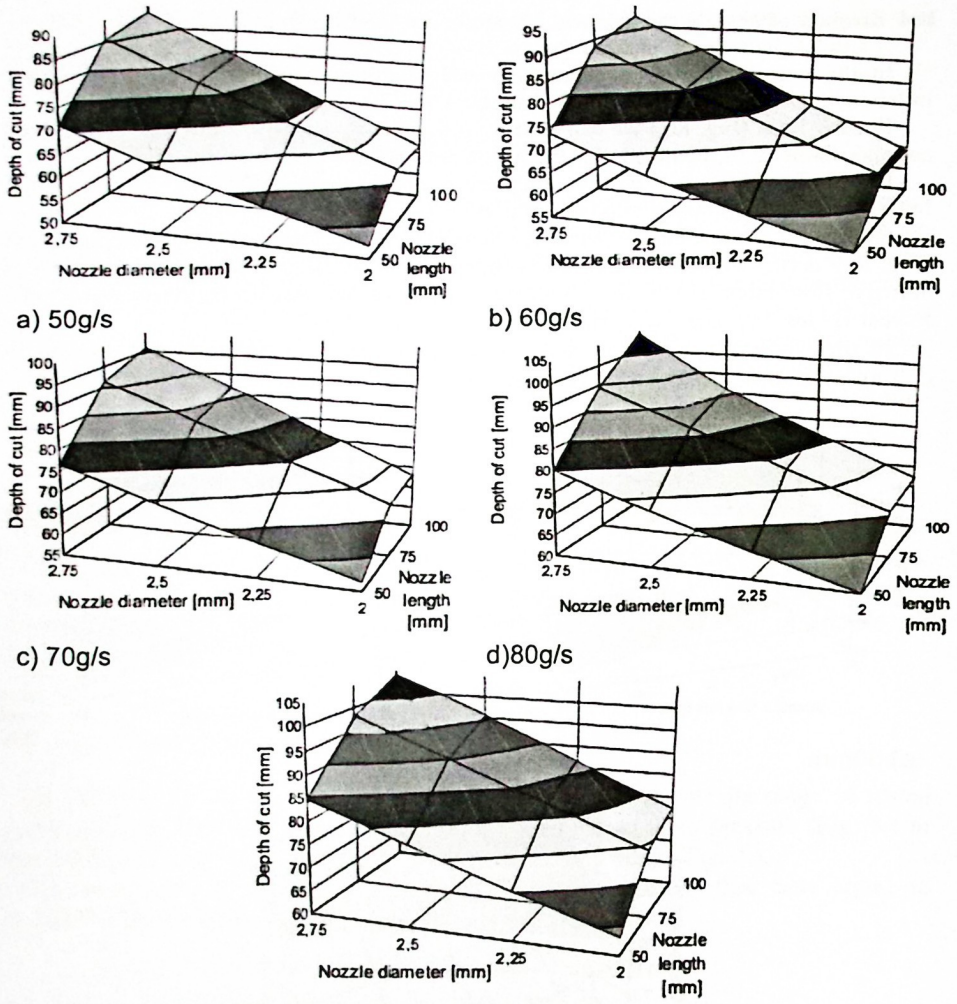


Fig. 9. Simulation of influence diameter and length of nozzle onto depth of cutting for variable abrasive flow rate.

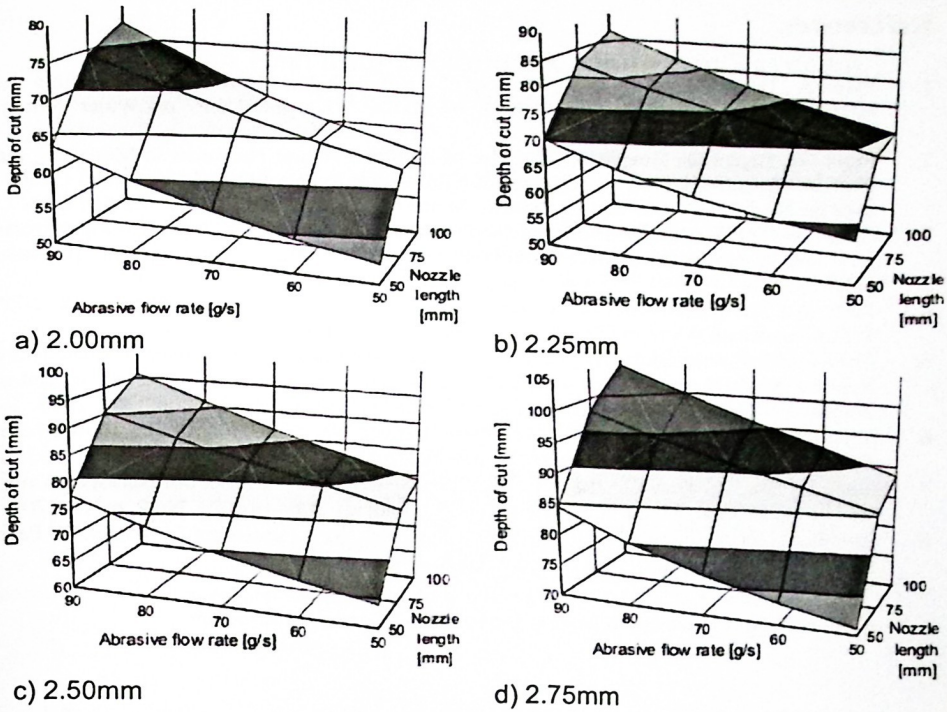


Fig. 10. Simulation of influence nozzle length and abrasive flow rate onto depth of cutting for variable nozzle diameter.

5 Summary

On the bases of analyses using neural networks we can conclude that the optimum cutting parameters from the perspective of maximizing cutting depth are as follows:

- Pressure p=28MPa,
- Traverse Speed V=1mm/s,
- Abrasive Flow Rate not less than 70g/s,
- Length of nozzle l=100mm,
- Diameter of nozzle ϕ=2.75mm

Neural networks are a very good tool for simulating abrasive cutting jet. The next step will be a comparison of results obtained from the simulation with actual cutting.

References

1. Perec A., Dżuga G., Ruzylo M.: The Use of Artificial Neural Networks in Modelling the Highpressure, Waterjet Cutting Process of Marble. 18th Int. Conf. on Water Jetting. Gdansk 2006.
2. Perec A., Dzuga G., Ruzylo M: The Use of Artificial Neural Networks in Modelling the Highpressure, Suspensive Waterjet Cutting Process of Syenite. First International Conference on Neural Networks and Associative Memories 2006 (NNAM 2006). Mexico City.
3. Perec A.: The Simulation Model of Highpressure, Suspension Waterjet Cutting Process with Use of the Artificial Neural Network. 2007 WJTA Amer. Waterjet Conf. Houston, USA 2007.
4. Perec A.: The Effectivenes of Hydroabrasive Suspensive Jet Cutting of the Rock. 2005 WJTA American Waterjet Conference. Houston, Texas, USA 2005, Paper 4B-1.
5. Srinivasu D.S., Ramesh B N. Srinivasa Y.G. Louis H., Peter D., Versemann R: Genetically Evolved Artifcial Neural Networks Built with Sparse Data for Predicting Depth of Cut in Abrasive Waterjet Cutting, 2005 WJTA American Waterjet Conf. Houston, USA 2005.
6. Vijay M.M., 1989, Evalution of Abrasive-Entreined Water Jets for Sloting Hard Rocks. 5th American Water Jet Conference. Toronto, Canada.
7. Yang L., Peng Z., Tang C., Zhang F.: Artificial Neural Network Model of Abrasive Waterjet Cutting Process, Mechanical Science and Technology 2004, Vol.23, No.2, p.218-220.
8. Zurada J..: Introduction to Artificial Neural Systems" West Publishing Company, St. Paul, New York, Los Angeles, San Francisco, USA. 1992.

Induction Learning Techniques Applied to Bayesian Networks Optimization

P. Britos, P. Felgaer, D. Rodríguez and R. García-Martínez

Software & Knowledge Engineering Center. Graduate School. Buenos Aires Institute of Technology
Intelligent Systems Laboratory. School of Engineering. University of Buenos Aires

rgm@itba.edu.ar

(Paper received on July 07, 2007, accepted on September 1, 2007)

Abstract: A bayesian network is a directed acyclic graph in which each node represents a variable and each arc a probabilistic dependency, they are used to provide: a compact form to represent the knowledge, and flexible methods of reasoning. Obtaining a bayesian network from data is a learning process that is divided in two steps: structural learning and parametric learning. In this paper, we define an automatic learning method that optimizes the bayesian networks applied to classification, using a hybrid method of learning that combines the advantages of the induction techniques of the decision trees (TDIDT - C4.5) with those of the bayesian networks. the resulting method is applied to prediction in health domain.

1 Introduction

The learning can be defined as "any process through as a system improves its efficiency". The ability to learn is considered as a central characteristic of the "intelligent systems" [Fritz *et al.*, 1989; García-Martínez & Borrajo, 2000], and for this a lot of effort and dedication was invested in the investigation and the development of this area. The development of the knowledge based systems motivated the investigation in the area of the learning with the purpose of automating the process of knowledge acquisition which considers one of the main problems in the construction of these systems.

The data mining [Perichinsky & García-Martínez, 2000; Perichinsky *et al.*, 2000; Perichinsky *et al.*, 2001; Perichinsky *et al.*, 2003] are the set of techniques and tools applied to the non-trivial process of extract and present/display implicit knowledge, previously unknown, potentially useful and humanly comprehensible, from large data sets, with object to predict of automated form tendencies and behaviors; and to describe of automated form models previously unknown, [Chen *et al.*, 1996; Mannila, 1997; Piatetski-Shapiro *et al.*, 1991]. The term Intelligent data mining, [Evangelos & Han, 1996; Michalski *et al.*, 1998] is the application of automatic learning methods, [Michalski *et al.*, 1983; Holsheimer & Siebes, 1991], to discover and enumerate present patterns in the data. For these, they were developed a great number of methods of analysis of data based on the statistic [Michalski *et al.*, 1982]. In the time

© H. Sossa, R. Barrón and E. Felipe (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 28, 2007, pp. 225-234

in which the amount of information stored in the databases was increased, these methods began to face problems of efficiency and scalability and is here where appears the concept of data mining. One of the differences between a traditional analysis of data and the data mining are that first it supposes that the hypotheses already are constructed and validated against the data, whereas the second supposes that the patterns and hypotheses automatically are extracted of the data.

The tasks of the data mining can be classified in two categories: descriptive data mining and predictive data mining [Piatetsky-Shapiro *et al.*, 1996; Han, 1999]; some of the most common techniques of data mining are the decision trees (TDIDT), the production rules and neuronal networks. On the other hand, an important aspect in the inductive learning, is the one to obtain a model that represents the knowledge domain and that is accessible for the user, in particular, is important to obtain the dependency data between the variables involved in the phenomenon, in the systems where it is desired to predict the behavior of some unknown variables based on certain known variables, a representation of the knowledge that is able to capture this information on the dependencies between the variables is the bayesian networks [Cowell *et al.*, 1990; Ramoni & Sebastiani, 1999].

A bayesian network is a directed acyclic graph in which each node represents a variable and each arc a probabilistic dependency, in which specifies the conditional probability of each variable given its parents; the variable at which it points the arc is dependent (cause-effect) of the variable in the origin of this one. The topology or structures of the network gives information on the probabilistic dependencies between the variables but also on conditional independences of a variable (or set of variables) given another or other variables, these independences simplify the representation of the knowledge (less parameters) and the reasoning (propagation of the probabilities).

Obtaining a bayesian network from data is a learning process that is divided in two phases: the structural learning and the parametric learning [Pearl, 1988]. First of them, consists of obtaining the structure of the bayesian network, that means, the relations of dependency and independence between the involved variables. The second phase has the purpose of obtain the a priori and conditional probabilities from a given structure.

The bayesian networks [Pearl, 1988] are used in diverse areas of application like medicine [Beinlich *et al.*, 1989], sciences [Bickmore & Timothy, 1994; Breese & Blake, 1995], and economy [Ezawa *et al.*, 1995]. They provide a compact form to represent the knowledge and flexible methods of reasoning -based on the probabilistic theories- able to predict the value of non-observed variables and to explain the observed ones. Some characteristics of the bayesian networks are that they allow to learn dependency and causality relations, they allow to combine knowledge with data [Heckerman *et al.*, 1995; Diaz & Corchado, 1999] and they can handle incomplete databases [Heckerman, 1995; Heckerman & Chickering, 1996; Ramoni & Sebastiani, 1996].

The bayesian networks are designed to find the dependence and independence relations between all the variables that conform the study domain, this allows to make predictions on the behavior of anyone of the unknown variables based on the values of the well-known variables; this estimates that any variable of the database can behave as incognito or as evidence according to the case.

Many practical tasks can be reduced to classification problems: medical diagnosis and pattern recognition are only two examples.
The bayesian networks can make the classification task -a particular case of prediction- that it is characterized to have a single variable of the database (class) that is desired to predict, whereas all the others are the data evidence of the case that is desired to classify. A great amount of variables in the database can exist; some of them directly related to the class variable but also other variables that have not direct influence on the class.
In this work, a method of automatic learning is defined that helps in the pre-selection of variables, optimizing the configuration of the bayesian networks in classification problems.

2 Methodology

In order to solve the problem of the bayesian networks applied to the classification task, in this work we use a hybrid learning method that combines the advantages of the induction techniques of the decision trees (TDIDT – C4.5) with those of the bayesian networks. For it, we integrate to the process of structural and parametric learning of the bayesian networks, a previous process of pre-selection of variables. In this process, it is chosen from all the variables of the domain, a subgroup with the purpose of generating the bayesian network for the particular task of classification and this way, optimizing the performance and improving the predictive capacity of the network.
The method for structural learning of bayesian networks is based on the algorithm developed by Chow and Liu (1969) to approximate a probability distribution by a product of probabilities of second order, which corresponds to a tree. The joint probability of variables can be represented like:

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i) P(X_i | X_{j(i)}) \quad (1)$$

where $X_{j(i)}$ it is the cause or parent of X_i.

Consider the problem like one of optimization and it is desired to obtain the structure of the tree that comes near more to the "real" distribution. A measurement of the difference of information between the real distribution (P) and the approximate one (P^*) is used:

$$I(P, P^*) = \sum_x P(X) \log(P(X) / P^*(X)) \quad (2)$$

Then the objective is to minimize I. A function based on the mutual information between pairs of variables is defined as:

$$I(X_i,X_j)=\sum_x P(X_i,X_j)\log P(X_i,X_j)/P(X_i)P(X_j)) \qquad (3)$$

Chow (1968) demonstrates that the information difference is a function of the negative of the sum of the mutual information (weights) of all the pairs of variables that constitute the tree. Reason why to find the more similar tree is equivalent to find the tree with greater weight. Based on that, the algorithm to determine the optimal bayesian network from data is the following one:

1. Calculate the mutual information between all the pairs of variables (n (n - 1) / 2).
2. Sort the mutual information in descendent order.
3. Select the arc of greater value as the initial tree.
4. Add the next arc while it does not form cycles. If it is thus, reject.
5. Repeat (4) until all the variables are included (n - 1 arcs).

Rebane and Pearl (1989) extended the algorithm of Chow and Liu for poly-trees. In this case, the joint probability is:

$$P(X)=\prod_{i=1}^{n} P(X_i \mid X_{j1(i)},X_{j2(i)},...,X_{jm(i)}) \qquad (4)$$

where $\{X_{j1(i)},X_{j2(i)},...,X_{jm(i)}\}$ is the set of parents for the variable X_i.

In order to compare the results obtained when applying the complete bayesian networks (RB-Complete) and the preprocessed bayesian networks with induction algorithms C4.5 (RB-C4.5), we used the databases "Cancer" and "Cardiology" obtained at the Irving Repository of Machine Learning databases of the University of California [Murphy & Aha] and the database "Dengue" obtained at the University of Buenos Aires [Carbajo *et al.*, 2003].
Table 1 summarizes these databases in terms of amount of cases, classes, variables (excluding the classes), as well as the amount of resulting variables of the preprocessing with the induction algorithm C4.5.
The methodology used to carry out the experiments with each one of the evaluated databases, is detailed next.

1. Divide the database in two. One of control or training (approximately 2/3 of the total database) and the another one of validation (with the remaining data)
2. Process the control database with the induction algorithm C4.5 to obtain the subgroup of variables that will conform the RB-C4.5
3. Repeat for 10%, 20%, ..., 100% of the control database
   3.1. Repeat 30 times, by each iteration

3.1.1. Take randomly X% from the control database according to the percentage that corresponds to the iteration
3.1.2. With that subgroup of cases of the control database, make the structural and parametric learning of RB-Complete and the RB-C4.5
3.1.3. Evaluate the predictive power of both networks using the validation database
3.2. Calculate the average predictive power (from the 30 iterations)
4. Graph the predictive power of both networks (RB-Complete and RB-C4.5) based on the cases of training

The step (1) of the algorithm makes reference to the division of the database in the control and the validation ones. In most cases, the databases obtained from the mentioned repositories were already divided.
For the pre-selection of variables by the induction algorithms C4.5 of the step (2), we introduced each one of the control databases in a decision trees TDIDT generating system. From there, we obtained the decision trees that represent each one of the analyzed domains. The variables that integrate this representation conform the subgroup that were considered for the learning of the preprocessed bayesian networks.
Next (3) a ten iteration process begins, in each one of these iterations processed 10%, 20%, 100% of the control database for the networks structural and parametric learning. This way, could be analyzed not only the difference in the predictive capacity of the networks, but also how evolved this capacity when we learn with greater amount of cases.
The objective of the repetitive structure of the step (3.1) is to minimize the accidental results that do not correspond with the reality of the model in study. It is managed to minimize this effect, taking different data samples and average the obtained values.
In the steps (3.1.x) it is made the structural and parametric learning of the RB-Complete and the RB-C4.5 from the subgroup of the control database (both networks are obtained from the same subgroup of data). Once obtained the network, it is come to evaluate the predictive capacity with the validation databases. This database is scan and for each row, all the evidence variables are instantiated and it is analyzed if the inferred class by the network corresponds with the indicated one in the file. Since the bayesian network does not make excluding classifications (it means that it predicts for each value of the class the probability of occurrence), is considered like the inferred class, the class with the greater probability. The predictive capacity corresponds to the percentage of cases classified correctly respect to the total evaluated cases.
In the point (3.2) it is calculated the predictive power of the network, dividing the obtained values through all the made iterations.
Finally, in the step (4) it is come to graph the predictive power average of both bayesian networks based on the amount of training cases.

Database	Variables	Variables C4.5	Classes	Control cases	Validation cases	Total cases
Cancer	9	6	2	500	199	699

Cardiology	6	4	2	64	31	95
Dengue	11	5	4	1.414	707	2.121

Table 1 – Databases description

3 Results

The experimental results were obtained by the application of the methodology previously mentioned to each one of the test databases.
As it can be observed in Figure 1 ("Cancer" domain), the predictive power of the RB-C4.5 is superior to the one of RB-Complete throughout all its points. Also, it is possible to observe how this predictive capacity is increased, almost always, when it takes more cases of training to generate the networks. Finally, it is observed that from the 350 cases of training the predictive power of the networks become stabilized reaching its maximum level.
When analyzing the graph of Figure 2 corresponding to the database "Cardiology", also an improvement on the RB-C4.5 can be observed respect to RB-Complete. Although the differences between the values obtained with both networks are smaller that in the previous case, the hybrid algorithm presents a better approach to the reality that the other one. It is important to emphasize that in this case, the improvement level is minimize when the set of cases used for the learning process is increased.
For the database "Degue" corresponding to Figure 3, an improvement in the predictive power of the proposed network is observed. The RB-C4.5 makes the classification with a 10% better precision than the other network.

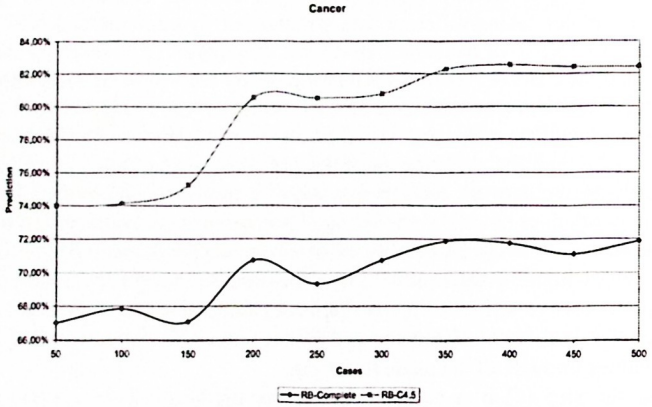


Figure 1 - Graph of the predictive power for the database "Cancer"

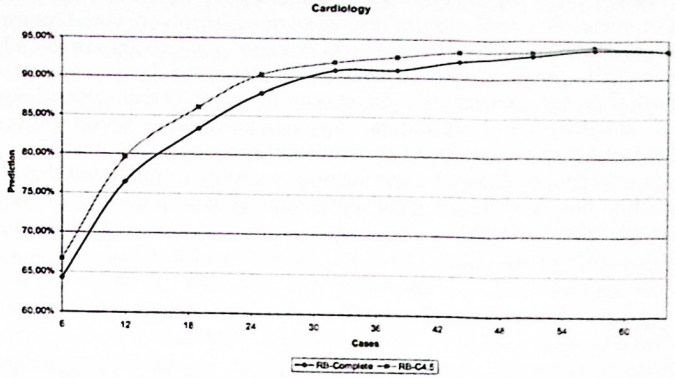


Figure 2 - Graph of the predictive power for the database "Cardiology"

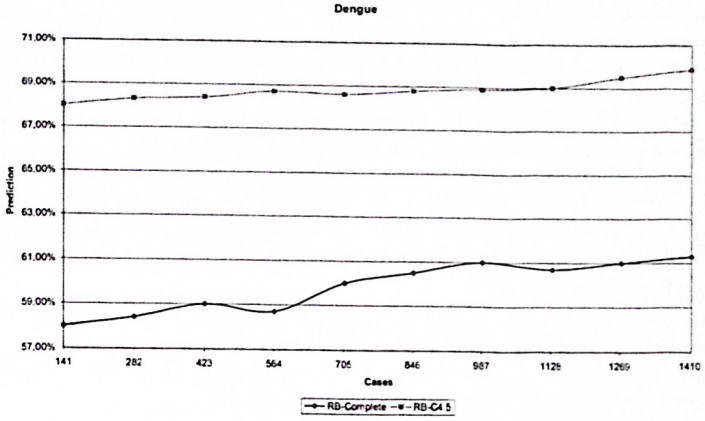


Figure 3 - Graph of the predictive power for the database "Dengue"

4 Discussion and Conclusions

As it is possible to observe, all the graphs that represent the predictive power based on the amount of cases of training are increasing. This phenomenon occurs independently of the domain of data used and the evaluated method (RB-Complete or RB-C4.5). Of the analysis of the results obtained in the experimentation, we can (experimentally) conclude that the learning hybrid method used (RB-C4.5) generates an improvement in the predictive power of the network with respect to the obtained one without making the preprocessing of the variables (RB-Complete).

In another aspect, the RB-C4.5 has a lesser amount of variables (or at the most equal) that RB-Complete, this reduction of the amount of involved variables produces a simplification of the analyzed domain, which carry out two important advantages; first, they facilitate the representation and interpretation of the knowledge removing parameters that do not concern on a direct way to the objective (classification task). Second, it simplifies and optimizes the reasoning task (propagation of the probabilities) which originates the improvement of the processing speed.
In conclusion, from the obtained experimental results, we concluded that the hybrid learning method proposed in this paper optimizes the configurations of the bayesian networks in classification tasks.

5 References

1. Beinlich, I.A., Suermondt, H.J., Chavez, R.M., Cooper, G.F. (1989). *The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks*. In proceedings of the 2nd European Conference on Artificial Intelligence in Medicine.
2. Bickmore, Timothy W. (1994). *Real-Time Sensor Data Validation*. NASA Contractor Report 195295, National Aeronautics and Space Administration.
3. Breese, John S., Blake, Russ (1995). *Automating Computer Bottleneck Detection with Belief Nets*. Proceedings of the Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Francisco, CA, pp 36-45.
4. Carbajo, A., Curto, S., Schweigmann, N. (2003). *Distribución espacio-temporal de Aedes aegypti (Diptera: Culicidae). Su relación con el ambiente urbano y el riesgo de transmisión del virus dengue en la Ciudad de Buenos Aires*. Departamento de Ecología, Genética y Evolución. Facultad de Ciencias Exactas y Naturales. Universidad de Buenos Aires
5. Chen, M., Han, J., Yu, P. (1996). *Data mining: An overview from database perspective*. IEEE Transactions on Knowledge and Data Eng.
6. Cowell, R., Dawid, A., Lauritzen, S., Spiegelhalter, D. (1990). *Probabilistic Networks and Expert Systems*. Springer, New York, NY.
7. Diaz, F., Corchado, J.M. (1999). *Rough sets bases learning for bayesian networks*. International workshop on objetive bayesian methodology, Valencia, Spain.
8. Evangelos, S., Han, J. (1996). *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. Portland, EE.UU.
9. Ezawa, Kazuo J., Schuermann, Til (1995). *Fraud/Uncollectible Debt Detection Using a Bayesian Network Based Learning System: A Rare Binary Outcome with Mixed Data Structures*. Proceedings of the Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Francisco, CA, pp 157-166.
10. Fritz, W., García-Martínez, R., Rama, A., Blanqué, J., Adobatti, R., Sarno, M. (1989). *The Autonomous Intelligent System*. Robotics and Autonomous Systems. Elsevier Science Publishers. Holanda. Volumen 5. Número 2. Páginas 109-125.
11. García-Martínez, R., Borrajo, D. (2000). *An Integrated Approach of Learning, Planning and Executing. Journal of Intelligent and Robotic Systems*. Volumen 29, Número 1, Páginas 47-78. Kluwer Academic Press.

12.Han, J. (1999). *Data Mining.* Urban and Dasgupta (eds.), Encyclopedia of Distributed Computing. Kluwer Academic Publishers.
13.Heckerman, D., Chickering, M., Geiger, D. (1995). *Learning bayesian networks, the combination of knowledge and statistical data.* Machine learning 20: 197-243
14.Heckerman, D. (1995). *A tutorial on learning bayesian networks.* Technical report MSR-TR-95-06, Microsoft research, Redmond, WA.
15.Heckerman, D., Chickering, M. (1996). *Efficient approximation for the marginal likelihood of incomplete data given a bayesian network.* Technical report MSR-TR-96-08, Microsoft Research, Microsoft Corporation.
16.Holsheimer, M., Siebes, A. (1991). *Data Mining: The Search for Knowledge in Databases.* Report CS-R9406, ISSN 0169-118X, Amersterdam, The Netherlands.
17.Mannila, H. (1997). *Methods and problems in data mining.* In Proc. of International Conference on Database Theory, Delphi, Greece.
18.Michalski, R.S., Baskin, A.B., Spackman, K.A. (1982). *A Logic-Based Approach to Conceptual Database Analysis.* Sixth Annual Symposium on Computer Applications on Medical Care, George Washington University, Medical Center, Washington, DC, EE.UU.
19.Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (1983). *Machine learning I: An AI Approach.* Morgan Kaufmann, Los Altos, CA.
20.Michalski, R.S., Bratko, I., Kubat, M. (1998). *Machine Learning and Data Mining, Methods and Applications.* John Wiley & Sons Ltd, West Sussex, England.
21.Murphy, P.M., Aha, D.W. *UCI Repository of Machine Learning databases.* Machine-readable data repository, Department of Information and Computer Science, University of California, Irvine.
22.Pearl, J. (1988). *Probabilisic reasoning in intelligent systems.* Morgan Kaufmann, San Mateo, CA.
23.Perichinsky, G., García-Martínez, R. (2000). *A Data Mining Approach to Computational Taxonomy.* Proceedings del Workshop de Investigadores en Ciencias de la Computación. Páginas 107-110. Editado por Departamento de Publicaciones de la Facultad de Informática. Universidad Nacional de La Plata. Mayo.
24.Perichinsky, G., García-Martínez, R., Proto, A. (2000). *Knowledge Discovery Based on Computational Taxonomy And Intelligent Data Mining.* CD del VI Congreso Argentino de Ciencias de la Computación. (\cacic2k\cacic\sp\is-039\IS-039.htm). Ushuaia. Octubre 2 al 6.
25.Perichinsky, G., García-Martínez, R., Proto, A., Sevetto, A, Grossi, D. (2001). *Data Mining: Supervised and Non-Supervised Intelligent Knowledge Discovery.* Proceedings del II Workshop de Investigadores en Ciencias de la Computación. Mayo. Editado por Universidad Nacional de San Luis en el CD Wicc2001:\Wiccflash\Areas\IngSoft\Datamining.pdf
26.Perichinsky, G., Servetto, A., García-Martínez, R., Orellana, R., Plastino, A. (2003). *Taxomic Evidence Applying Algorithms of Intelligent Data Minning Asteroid Families.* Proceedings de la International Conference on Computer Science, Software Engineering, Information Technology, e-Bussines & Applications. Pág. 308-315. Río de Janeiro (Brasil). ISBN 0-9742059-3-7.
27.Perichinsky, G., Servente, M., Servetto, A., García-Martínez, R., Orellana, R., Plastino, A. (2003). *Taxonomic Evidence and Robustness of the Classification*

Applying Intelligent Data Mining. Proceedings del VIII Congreso Argentino de Ciencias de la Computación. Pág. 1797-1808.

28.Piatetski-Shapiro, G., Frawley, W.J., Matheus, C.J. (1991). *Knowledge discovery in databases: an overview*. AAAI-MIT Press, Menlo Park, California.

29.Piatetsky-Shapiro, G., Fayyad, U.M., Smyth, P. (1996). *From data mining to knowledge discovery*. AAAI Press/MIT Press, CA.

30.Ramoni, M., Sebastiani, P. (1996). *Learning bayesian networks from incomplete databases*. Technical report KMI-TR-43, Knowledge Media Institute, The Open University.

31.Ramoni, M., Sebastiani, P. (1999). *Bayesian methods in Intelligent Data Analysis. An Introducction*. Pages 129-166. Physica Verlag, Heidelberg.

Author Index
Índice de Autores

Barrón, Ricardo 69
Baruch, Ieroham S. 183, 203
Britos, P. 225
Chakraborty, Debrup 127
Cuevas, Erik 173
De Jesús, Orlando 3
De la Cruz, Irving P. 183
Fathy, Mahmood 81
Felgaer, P. 225
Flores, Eber J. 161
García, R. 225
Garza, Luis E. 139, 193
Godoy, Salvador 93
Gomez, Eduardo 39
Hagan, Martin 3
Jiřina, jr., Marcel 29
Jiřina, Marcel 29
Kim, Man-Sun 151
López, Fabiola 161
Mariaca, Carlos R. 183, 203
Mejía, Manuel 115
Morales, Rubén 139
Mozayani, Nasser 81
Nakano, Mariko 105
Nieto, Juan P. 139
Olivares, Jesus 105
Pazienza, Giovanni 39
Perec, Andrzej 215
Perez, Hector 105
Perez, Marco 173
Rodríguez, D. 225
Rojas, Raúl 173
Ruffo, Geovanna 193
Sánchez, G. 69
Sanchez, Gabriel 105
Segura, Enrique 15
Shahbazi, Hamed 81
Soryani, Mohsen 81
Sossa, Humberto 59, 69
Sucar, Enrique 115
Torres-Treviño, L. 49
Toscano, Linda 69
Vázquez, Eduardo F. 127
Vázquez, Roberto 59, 93
Vilariño, Darnes 161
Vilasís, Xavier 39
Zaldivar, Daniel 173

Editorial Board of the Volume

Comité Editorial del Volumen

Akira Hirose
Alejandro Rosete
Alessandro E.P. Villa
Alexander Poznyak
Alfredo Weitzenfeld
Amir Hussain
Andre Ponce de Leon
Andrzej Cichocki
Andrés Pérez Uribe
Ángel Kuri
Anne Guérin
Antonio Bahamonde
Antony Satyadas
Arturo Hernández
Bart Kosko
Bernard Widrow
Bert Kappen
Brijesh Verma
Bruno Gas
Bruno Jammes
Carlos A. Reyes
Carlo F. Morabito
Changyin Sun
Changjiu Zhou
Chilukuri K. Mohan
Christian Jutten
Christophe García
Colin Fyfe
Danil Prokhorov
Danilo Manidic
David A. Elizondo
David Hoyle
Delian Wang
Derek Partridge
Derong Liu
Dionisio Suárez
Du Zhang
Edgar Sánchez
Eduardo Gómez
Elmar W. Lang
Exiquio Leyva
Fabián Theis
Francisco Cervantes
Francisco Cuevas
Francesco Masulli
Francesco Morabito
Franz Wotawa
Georgios Anagnostopoulos
Giacomo Indiveri
Gustavo Arroyo
Héctor Pérez
Huaguang Zhang
Ieroham S. Baruch
Janusz Kacprzyk
Janusz Kacprzyk
Jeanny Hérault
Jesús Favela
Jesus Cid-Sueiro
José A. Ruz Hernández
Jesús G. Figueroa
Johan Suykens
John Qiang Gan
John Taylor
José Dorronsoro
Jose Principe
Juan Arturo Nolazco
Juan Flores
Juan Pedro Febles
Klaus Obermayer
Khurshid Ahmad
Kunihiko Fukushima
Liang Chen
Lipo Wang
Luiza de Macedo
Marcilio C. P. De Souto
Manuel Graña

Manuel Mejía Lavalle
Marley Vellasco
Marcos Eduaro Valle
Michael Hasselmo
Michael Small
Miguel Á. Garay Garcell
Mohamed Chetouani
Nikolaos Bourbakis
Nikola Kasabov
Oscar Castillo
Okyay Kaynak
Paolo Lisboa
Patricia Melin
Patricia Rayón
Pedro Y. Piñero Pérez
Peter Andras
Péter Érdi
Peter Sussner
Peter Szolgay
Raúl Gardúño Ramírez
Raúl Rojas
René V. Mayorga
Richard Duro
Rogelio Soto
Sanqing Hu
Sergi Bermejo
Sheng Chen
Shinichiro Omachi
Soo-Young Lee
Stan Gielen
Stefanos Kollias
Teresa Ludermir
Witold Pedrycz
Wlodzislaw Duch
Xavier Vilasis Cardona
Xiao Hui Liu
Xinghuo Yu
Yohsuke Kinouchi
Zeng-Guang Hou

Impreso en los Talleres Gráficos
de la Dirección de Publicaciones
del Instituto Politécnico Nacional
Tresguerras 27, Centro Histórico, México, D.F.
Noviembre de 2007.
Printing 500 / Edición 500 ejemplares.

Neural Networks, Associative Memories together with Fuzzy Systems and Genetic Algorithms are part of the so called soft computing techniques aimed at providing a machine elements of human-like behaviour such as pattern recognition, learn by examples, function approximation, and so on. In addition to creating human-like-behaving machines, artificial neural networks and associative memories provide a very powerful platform for solving a wide range of complex optimisation problems.

This volume presents 21 carefully selected original research papers on the internal art and craft of neural networks and associative memories: its theoretical foundations, specific techniques and research methodologies. The papers are structures into eight thematic fields:

- Feature Extraction and Dimensionality Reduction.
- Image Processing and Applications.
- Neural Networks and Associative Memories.
- Real World Applications.

ISSN: 1870-4069
www.cic.ipn.mx/rcs
www.ipn.mx

SEP

INSTITUTO POLITÉCNICO NACIONAL
"La Técnica al Servicio de la Patria"