RESEARCH IN COMPUTING SCIENCE

ISSN: 1870-4069



Special issue: Neural Networks and Associative Memories

Juan Humberto Sossa Azuela
Ricardo Barrón Fernández
(Eds.)

Vol. 21

RCS
Research in Computing Science

Special Issue: 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 (France)
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 21,** Noviembre, 2006. 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 21,*** *November, 2006. 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, 2006, in the IPN Graphic Workshop – Publication Office.*

Volume 21

Volumen 21

Special Issue: Neural Networks and Associative Memories

Volume Editors:
Editores del Volumen

Juan Humberto Sossa Azuela
Ricardo Barrón Fernández

Instituto Politécnico Nacional
Centro de Investigación en Computación
México 2006

ISSN: 1665-9899

Copyright © 2006 Instituto Politécnico Nacional 2006
Copyright © 2006 Instituto Politécnico Nacional

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

Indexada en LATINDEX
Indexed in LATINDEX

Printing: 500
Impresiones: 500

Printed in Mexico
Impreso en México

Preface

Neural Networks and Associative Memories are tools that have been used for many years. Their theoretical foundations are now well established. Its 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 word applications. This special volume is structured into seven thematic fields representative of some of the main current areas of interest within the NN and AMS communities:

- Algorithms and Learning,
- Associative Memories,
- Data Analysis, Pattern Recognition and Self-Organization,
- Neural Control and Robotic Applications,
- Real World Applications,
- Support Vector Machines, and
- Vision and Image Processing.

A total of 51 full papers from 8 different countries were submitted for evaluation. Each submission was reviewed by three independent members of the Editorial Board of the volume. This volume contains revised versions of 25 papers, selected for publication after thorough evaluation. Thus the acceptance rate was of 49.01%.

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

November 2006

Table of Contents

Índice

Page/Pág.

Algorithms and Learning

Increasing Boltzmann Machine Learning Speed and Accuracy with the High Order Decimation Method 3
Enric Farguell, Eduardo Gómez and Ferran Mazzanti

Bounded Weights Trajectories for Neural Network State Estimation 13
Isaac Chairez, Alexander Poznyak and Tatyana Poznyak

Hepatitis C Viral Load Control Applying Hamilton-Jacobi And Sliding Mode Techniques 25
Ramón F. Miranda, Isaac Chairez and Agustín Cabrera

Associative Memories

Associative Memory: Information and Topology 39
David Dominguez, Kostadin Koroutchev, Eduardo Serrano and Francisco Rodríguez

Alternative Algorithm for Multi-Layered Morphological Networks 49
Ricardo Barrón, Humberto Sossa and Benjamín Cruz

Efficient Codebooks for Visual Concept Recognition by ASSOMs Activation 61
Grégoire Lefebvre and Christophe Garcia

Data Analysis, Pattern Recognition and Self-Organization

Analysis and Classification of Car Engine Sounds to Diagnose Failures with the Help of Neural Networks 75
Luis A. Méndez and Orión F. Reyes

Personalizing Google's Search Results using Naïve Bayesian Probabilistic Model 87
Srinivasa K G, Srinivas Kumar M, Vinay T C, Venugopal K R and Lalit M Patnaik

PM10 Classification with SOM Neural Networks for Air Quality Levels Estimation 97
Maria G. Cortina, J. Miguel Barrón, Sergio Ledesma, D. Andina and A. Vega

Effect of Long-delay Connections to Realize a Self-Organization of Firing Patterns that Vary Smoothly in Space 105
Toshio Akimitsu, Yoichi Okabe and Akira Hirose

Neural Control and Robotic Applications

Local Optimal Control Based on Neural Network Observer States 117
Nadezhda Aguilar, Agustin Cabrera and Isaac Chairez

A Direct Torque Vector Neural Control of a Three Phase Induction Motor 131
Ieroham S. Barouh, Carlos R. Mariaca and Irving P. de la Cruz

Neural Network Model for Balancing a Biped Robot 141
Daniel Zaldivar, Erik Cuevas, Marco A. Pérez and Raul Rojas

RNN Identification and Control of Nonlinear Plants Using the Recursive Marquardt Algorithm 151
Ieroham Baruch, Saul Escalante and Carlos R. Mariaca

Forward Models and the Prediction of Undesired Situations 161
Bruno Lara, Juan Manuel Rendon and Marcos Capistran

Real World Applications

The Use of Artificial Neural Networks in Modeling the High-Pressure, Suspensive Waterjet Cutting Process of Syenite..... 173
Andrzej Perec, Grzegorz Dzuga and Mateusz Ruzylo

Physical Model-Free Observer for Nuclear Reactors using Differential Neural Networks with Sliding Mode Learning 181
José H. Pérez and Alexander Poznyak

Unusual Changes of Consumption Detection in Mobile Phone Users 195
Paola Britos, Hernan Grosser, Enrique Sierra and Ramon Garcia

Selecting Pedagogical Protocols Using SOM..... 205
Fernando Salgueiro, Zulma Cataldi, Paola Britos, Enrique Sierra and Ramon Garcia

Support Vector Machines

Drowsy Driver Detection Using Wavelets and Support Vector Machines 217
Rodolfo E. Ibarra, Miguel González, Ariel L. García, Neil Hernández and Jaime Mora

Isolated Character Recognition by using Support Vector Machines and Optimal Feature Vectors 227
Hermenegildo Cisneros, Ricardo Barrón and Benjamín Cruz

Heuristics to Reduce the Training Time of SVM Algorithm 237
Ariel L. García, Neil Hernández, Miguel González, Rodolfo E. Ibarra and Jaime Mora

Vision and Image Processing

Performance of Lossy Image Compression Coders on Textures 249
Osslan O. Vergara, Raúl Pinto and José Ruíz

SOFM Neural Network Texture Segmentation 259
Erik Cuevas, Daniel Zaldivar, Marco A. Pérez and Raul Rojas

Licence Plate Location and Recognition Using Decision Regions of Color with Genetic Algoritms and Neural Networks ... 269
Gilberto A. Apodaca and Patricia Rayón

Author Index 281

Algorithms and Learning

Increasing Boltzmann Machine Learning Speed and Accuracy with the High Order Decimation Method

Enric Farguell[1], Eduardo Gómez-Ramírez[2], and Ferran Mazzanti[3]

[1] Dept. d'Electrònica, EALS La Salle, Universitat Ramon Llull, Quatre Camins 2, E-08022 Barcelona, Spain
efarguell@salle.url.edu

[2] Postgrado e Investigación, Universidad La Salle México, Benjamín Franklin 47, Col. Hipódromo Condesa 06140 México D.F., México
egr@ci.ulsa.mx

[3] Departament de Física i Enginyeria Nuclear, Universitat Politècnica de Catalunya, Comte Urgell 187, E-08036 Barcelona, Spain
ferran.mazzanti@upc.edu

(Paper received on July 31, 2006, accepted on September 25, 2006)

Abstract. The Boltzmann Machine is a stochastic recurrent neural network with the ability to learn and extrapolate probability distributions. However, it has a serious drawback in the exhaustive computational cost involved at learning and simulation stages. Decimation [1, 2] was originally introduced as a way to overcome this problem. Despite its success, the method was too restrictive as it could only be used on sparsely connected Boltzmann Machines with stringent constraints on the connections between the units. The High Order Decimation technique proposed in this work is an extension of the previous method to any Boltzmann Machine with no restrictions on connections nor topology, and is based on the use of high order weights which incorporate additional degrees of freedom. An example on binary patterns is also presented, showing the improvement in efficiency against the performance of the original Boltzmann Machine and a suitably tuned perceptron.

1 Introduction

The Boltzmann Machine (BM) [3] is a stochastic and recurrent neural network based on the Hopfield model [4] with the ability to learn and extrapolate probability distributions. It is formally equivalent to a spin glass [5], thus providing a clear definition of its dynamic and static properties. On the other hand, the exhaustive computational cost implied by the model has prevented its use in most applications of practical interest.

Standard decimation [1, 2] was proposed as a method to reduce the total amount of computational power needed at the learning stage, reducing this process to the solution of a coupled set of nonlinear equations. However and as conceived originally, this set could only be easily solved for a very special and sparse kind of topologies known nowadays as decimatable [2] due to the lack

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 3-12

of enough degrees of freedom when applied to regular BMs. The High Order Decimation method explained in the following sections is able to deal with any Boltzmann Machine regardless of its topology, as it adds high order connections which incorporate the extra variables into the equations without a dramatic increase in complexity.

2 The Boltzmann Machine

2.1 Boltzmann Machine Dynamics

The Boltzmann Machine is a Hopfield-like neural network with dynamics inherited from the Simulated Annealing [6] optimization algorithm. It is governed by a cost function that is written in terms of its unit states $S_i = [-1, +1]$

$$E = -\frac{1}{2}\sum_{i,j} w_{ij}^{(2)} S_i S_j - \sum_i w_i^{(1)} S_i \ , \tag{1}$$

where $w_{ij}^{(2)} = w_{ji}^{(2)}$ is a symmetric weight connecting units S_i and S_j, while $w_i^{(1)}$ is a bias term connected to unit S_i. This functional is evaluated with the Simulated Annealing algorithm until the Boltzmann Machine reaches thermal equilibrium at the lowest temperature of the annealing schedule. However, the goal of the process is not to optimize this function but rather to reach a stationary probability distribution which in the end follows a Boltzmann law

$$p(E_\alpha) = \frac{e^{-E_\alpha/T}}{Z} \ , \tag{2}$$

$$Z = \sum_\gamma e^{-E_\alpha/T} \ , \tag{3}$$

where the index α denotes the state of the network and could be written as a string of $+1$ and -1 indicating the state of every unit in the network. Finally, T is the current temperature from the annealing schedule and Z is a normalization term usually known as the *partition function.*

2.2 Learning in Boltzmann Machines

The learning process in a BM is done by minimization of the Kullback-Leibler distance [7] between the actual probability distribution and the one to be learnt. For a BM with different input and output units this quantity reads

$$G = \sum_\alpha R_{\alpha|\gamma} \ln \frac{R_{\alpha|\gamma}}{P_{\alpha|\gamma}} \ , \tag{4}$$

where $P_{\alpha|\gamma}$ is the Boltzmann probability of finding an output state α when a state γ has been set in the input units and, consequently, depends on the weights

and biases of Eq. (2). $R_{\alpha|\gamma}$ is the desired probability distribution to be learnt for the same inputs and outputs, and is shown to the network in the form of an input-output training set. Since $G > 0$ for any $P_{\alpha|\gamma} \neq R_{\alpha|\gamma}$ and has a global minimum $G = 0$ when $P_{\alpha|\gamma} = R_{\alpha|\gamma}$, gradient descent is usually employed on Eq. (4) to find the update rule for both weights and biases [5] during learning. Thus

$$\Delta w_{\sigma}^{(m)} = -\eta \frac{\partial G}{\partial w_{\sigma}^{(m)}} \ , \tag{5}$$

where $w_{\sigma}^{(m)}$ is a weight term connecting the m units denoted by the general index σ. Hence, for $m = 2$ and $\sigma = i, j$, $w_{ij}^{(2)}$ is the two-unit weight connecting S_i, S_j. When $m = 1$, $\sigma = i$ and becomes the bias term $w_i^{(1)}$ associated to S_i. Calculating derivative from Eq. 5 will lead to expression

$$w_{\sigma}^{(m)} = \frac{\eta}{T} \left(\left\langle \prod_{\rho \in \sigma} S_{\rho} \right\rangle^{*} - \left\langle \prod_{\rho \in \sigma} S_{\rho} \right\rangle \right) \ , \tag{6}$$

where $< \ldots >^{*}$ and $< \ldots >$ denote expectation values of its arguments on the current probability distribution, while the star on the average indicates that the output units are *clamped* to the desired output for a given input. These moments have to be calculated for each vector of the training set, and are estimated in the Monte Carlo simulations or annealing process. Finally, η is a convergence parameter and T the lowest equilibrium temperature from the cooling schedule.

2.3 The High Order Boltzmann Machine

The High Order Boltzmann Machine (HOBM) [8, 9] is an extension to the original model where connections through more than two units are allowed. In this way, weights on a M-th order HOBM can connect up to M units, though not all feasible combinations may necessarily be implemented. An example of a high order weight connecting three units is shown in Fig. 1.

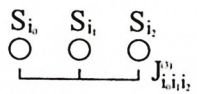


Fig. 1. Simplified notation for 3rd order connections

Of course the cost function or energy functional also changes as it has to allow for the inclusion of the new weights. The most natural extension is [9]

$$E = -\sum_{i_0} w_{i_0}^{(1)} S_{i_0} - \frac{1}{2!} \sum_{i_0, i_1} w_{i_0 i_1}^{(2)} S_{i_0} S_{i_1} - \frac{1}{3!} \sum_{i_0, i_1, i_2} w_{i_0 i_1 i_2}^{(3)} S_{i_0} S_{i_1} S_{i_2} - \ldots$$

$$= -\sum_{m=1}^{M}\left(\frac{1}{m!}\sum_{\sigma} w_{\sigma}^{(m)}\prod_{\rho\in\sigma} S_{\rho}\right) , \tag{7}$$

where m denotes the order of the weight and σ runs over the sets of m units actually connected in the network considered. Despite the change in E the dynamics remain the same, and a Boltzmann probability distribution corresponding to that energies is still reached at equilibrium.

Learning in a HOBM is carried out as in a standard BM, and so the Kullback-Leibler distance is still used to find weight updating expressions, which conform to Eq. (6) with the appropriate extension of the indexes displayed. Nevertheless, the evaluation of the expectation values in the Monte Carlo simulation becomes more and more involved with increasing m, as stated on Ref. [10]. The High Order Decimation method allows an analytical evaluation of these values, as it is explained in the following sections.

3 Standard Decimation of Boltzmann Machines

Decimation is a standard technique in statistical mechanics [11] that is used in this context as a procedure to reduce the number of weights and units on a Boltzmann Machine with stringer connectivity constraints [1,2]. Given a standard 2nd order BM with N units and tree topology, it is possible to build a smaller network with $n < N$ units that still produces the same expectation values of Eq. (6). This is accomplished by decimating any given S unit connected to no more than three other units S_i, S_j and S_k, regardless of whether they are clamped or not.

Decimation is better carried out when a new, temperature normalized weights are employed

$$J_{\sigma}^{(m)} = \frac{w_{\sigma}^{(m)}}{T} . \tag{8}$$

In its practical form, decimation of a unit S connected to other neurons by two-unit weights is accomplished when the corresponding terms in the partition function fulfill the relation

$$\sum_{S=-1}^{+1} e^{S\sum_i S_i J_i^{(2)}} = \sqrt{C} e^{\sum_{j<i} S_i S_j J_{ij}^{(2)}} . \tag{9}$$

This leads to the master equation

$$\ln\cosh\left(\sum_i S_i J_i^{(2)}\right) = J^{(0)} + \sum_{j<i} S_i S_j J_{ij}^{(2)} , \tag{10}$$

where $J^{(0)} = \ln(\sqrt{C}/2)$ is a normalization constant that is required to solve the system of equations. Notice that the goal of this procedure is to *remove* unit S from the network by changing the weights among the rest of the neurons in the

system. In this way the $J_i^{(2)}$ weights appearing on the lhs of the equations above stand for their values prior to the decimation, while the $J_{ij}^{(2)}$ weights on the rhs correspond to the new set of weights once the decimation has been carried out.

3.1 Decimation Equations

A simple example of high order decimation is the star–triangle transformation of Fig. 2. The new weights obtained from the decimation of the unit in the center of the star become

$$J_{ij}^{(2)} = \frac{1}{4} \ln \cosh \left(\frac{\left(J_i^{(2)} + J_j^{(2)} + J_k^{(2)}\right)\left(J_i^{(2)} + J_j^{(2)} - J_k^{(2)}\right)}{\left(J_i^{(2)} - J_j^{(2)} - J_k^{(2)}\right)\left(J_i^{(2)} - J_j^{(2)} + J_k^{(2)}\right)} \right) ,$$

$$J_{ik}^{(2)} = \frac{1}{4} \ln \cosh \left(\frac{\left(J_i^{(2)} + J_j^{(2)} + J_k^{(2)}\right)\left(J_i^{(2)} - J_j^{(2)} + J_k^{(2)}\right)}{\left(J_i^{(2)} - J_j^{(2)} - J_k^{(2)}\right)\left(J_i^{(2)} + J_j^{(2)} - J_k^{(2)}\right)} \right) ,$$

$$J_{jk}^{(2)} = \frac{1}{4} \ln \cosh \left(\frac{\left(J_i^{(2)} + J_j^{(2)} + J_k^{(2)}\right)\left(J_i^{(2)} - J_j^{(2)} - J_k^{(2)}\right)}{\left(J_i^{(2)} + J_j^{(2)} - J_k^{(2)}\right)\left(J_i^{(2)} - J_j^{(2)} + J_k^{(2)}\right)} \right) . \quad (11)$$

The serial association of Fig. 2b is a particular case of the previous result when $J_k^{(2)}$ is set to zero

$$J_{ij}^{(2)} = \frac{1}{2} \ln \left(\frac{\cosh\left(J_i^{(2)} + J_j^{(2)}\right)}{\cosh\left(J_i^{(2)} - J_j^{(2)}\right)} \right) . \quad (12)$$

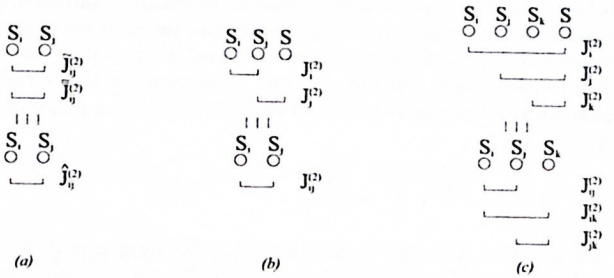


Fig. 2. Serial (a), parallel (b) and star-triangle association (c)

Finally, the parallel association of Fig. 2a is trivially seen to be the sum of the weights and is a simplification rule rather than a decimation one that is included for consistency reasons.

4 High Order Decimation

Decimation, as stated in the previous section, can not be used to simplify a unit connected to more than three other units. Hence, a structure like the one showed in Fig. 3 can not be decimated with the techniques explained so far.

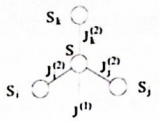


Fig. 3. Simplest non decimatable structure

Decimation fails to simplify these structures because the equations obtained are not compatible with one another. The High Order Decimation method adds enough high order terms for the system to become solvable. In order to explain why this happens, it is useful to take a look at the equations that describe the easiest non-solvable example, which consists of a unit S connected to three other units and a bias term as shown in Fig. 3

$$\ln\cosh\left(J^{(1)} + J_i^{(2)}S_i + J_j^{(2)}S_j + J_k^{(2)}S_k\right) = J^{(0)} + J_{ij}^{(2)}S_iS_j + J_{ik}^{(2)}S_iS_k + J_{jk}^{(2)}S_jS_kJ_{jk}^{(2)} \quad . \tag{13}$$

The complete set of equations born from this expression can not be solved for arbitrary values of $J^{(1)}, J_i^{(2)}, J_j^{(2)}$ and $J_k^{(2)}$. The resulting system of equations can only be solved if there are as many equations as unknown variables, provided that the equations are linearly separable. Hence, four variables are needed. If bias terms associated to units S_i, S_j and S_k are included, there is only one unknown left. A third order weight can then be used to account for this variable. This term will link these three units in the set of equations, and appears as $J_{ijk}^{(3)}$

$$\ln\cosh\left(J^{(1)} + \sum_i S_iJ_i^{(2)}\right) = J^{(0)} + \sum_i S_iJ_i^{(1)} + \sum_{j<i} S_iS_jJ_{ij}^{(2)} + \sum_{k<j<i} S_iS_jS_kJ_{ijk}^{(3)} \quad . \tag{14}$$

This set of equations is characterized by a Hadamard matrix [12]. Hadamard matrices have orthogonal rows and columns so the system has a solution that is unique. The same thing happens when an $N+1$-star groups of units is considered, where a central biased S unit is connected to units S_0 to S_{N-1}. In this case, weights up to N-th order have to be added. As a result, a fully connected N-th

order neural network is obtained. The total number of equations required to perform decimation is 2^N, as there are $\binom{N}{0} = 1$ normalization constants $J^{(0)}$, $\binom{N}{1}$ biases $J^{(1)}$, $\binom{N}{2}$ second order weights $J^{(2)}$, $\binom{N}{3}$ third order terms $J^{(3)}$ and so on, until the last $\binom{N}{N} = 1$ N-th order $J^{(N)}$ weight. This makes a total of 2^N variables. The process is schematically shown in Fig. 4, where the originally second order neural network on Fig. 4a is decimated to produce the result of Fig. 4b. It can be proved that this system has always an associated Hadamard matrix, meaning that there will always be a unique solution for a given structure.

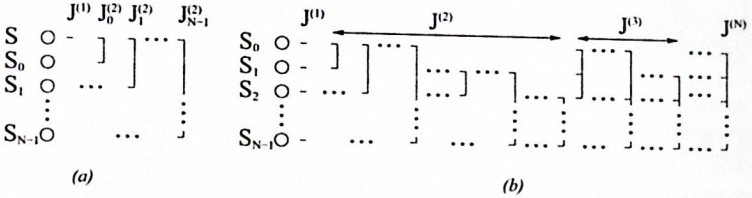


Fig. 4. Original (a) and decimated (b) structure. New weights $J^{(m)}$ connect as much as m different S_i units

When the decimation is to be carried out, all weights connected to the central unit S have to be included, and the corresponding equations have the following form

$$\ln\cosh\left(J^{(1)} + \sum_i S_i J_i^{(2)} + \sum_{j<i} S_i S_j J_{ij}^{(3)} + \ldots + \sum_\sigma J_\sigma^{(m)} \prod_{\rho\in\sigma, S\notin\rho} S_\rho\right) =$$
$$= J^{(0)} + \sum_i S_i J_i^{(1)} + \sum_{j<i} S_i S_j J_{ij}^{(2)} + \sum_{k<j<i} S_i S_j S_k J_{ijk}^{(3)} + \ldots \,. \tag{15}$$

Once the system is solved, a new Boltzmann Machine with one less unit is left, at the expense of it being highly connected due to the many different high order weights produced by the decimation reduction. In order to compute the needed m-th order correlations appearing in the weight update learning rule, the process is iterated until all but the required units are decimated, in such a way that the expectation values entering in the updates can be computed analytically as noted in Ref. [1, 2]. The whole process has to be repeated for every weight that has to be updated.

5 Results and Discussion

The high order decimation method has been tested against a perceptron and a traditional BM in a letter recognition dataset context, where the network has to recognize characters from a noisy source. A system of 24 letters written with a *Times New Roman* font is used, and each letter is represented by a 50x50 pixels binary image. While these neat characters are the ones to be learned, a set of 100 different images for each letter is generated by adding random noise which is implemented via bit negation. The amount of noise present is characterized by a parameter γ, which is proportional to the percentage of negated bits. Thus for example, 10% of bits are reversed when $\gamma = 10$. Once the learning using the previous patterns has been carried out, a new set of noisy characters is generated using the same procedure to test the network.

In practical terms, execution time is sped up when a momentum term characterized by an extra parameter α is added to the original gradient descent expression [13]

$$w^{(k+1)} = w^{(k)} - \eta(1-\alpha)\frac{\partial \mathcal{E}}{\partial w} + \alpha w^{(k-1)} \quad , \tag{16}$$

$\partial \mathcal{E}/\partial w$ denoting the partial derivative of the error. Since each neural network uses a different measure for the error, this expression may be thought as a generalization of the learning algorithm which has been adapted for the three networks used in this work.

5.1 Monte Carlo Implementation

Table 1 shows the amount of time and epochs (equal to the number of times weights are updated in a complete run of the learning algorithm) required by a Boltzmann Machine trained with the High Order Decimation method compared with results for the same network trained with the standard Monte Carlo algorithm using the above stated patterns for $\gamma = 20$. All calculations have been performed on a DELL workstation mounting a Pentium Xeon EMT64 with 2Mb of cache processor working at 3.0 GHz and equipped with 1.0 Gb DDR2 ECC RAM memory. As it can be seen, not only the High Order decimation performs *faster* but also requires less epochs to reach the desired result. This is due to the fact that every Monte Carlo simulation has an associated statistical error, and bringing that below a certain limit (imposed by the accuracy to be achieved) can be very expensive in computational terms. The network used in this calculation has 2500 input units (corresponding to the 50x50 pixel images used as input), 1 hidden and 5 output units. The training parameters were $\eta = 0.2$, $\alpha = 0.1$, maximum absolute error $\mid \partial \mathcal{E}/\partial w \mid = 0.05$ and maximum absolute initial random value for the weights $\mid w_0 \mid = 1.0$.

The relation *time/epoch* describes how long does it take to run a complete epoch in the simulation. As it can be seen from the table, Decimation performs considerably better in both aspects. A Decimation epoch is faster because there is no need to run a Simulated Annealing but only to solve a system of equations.

Table 1. Decimation method against Monte Carlo implementation

Algorithm	Mean epochs	Mean time/epoch (seconds)
Monte Carlo	45	586.68
High Order Decimation	11	4.39

On the other hand, it needs less epochs to end because it does not suffer from statistical errors as does a Monte Carlo simulation.

5.2 Perceptron

Finally, a comparison between the performance of the BM trained with the High Order Decimation method and a dual layer perceptron is presented. The comparison is made on the basis that both networks can provide a *full* solution to the problem at hand if enough learning instances are allowed. In fact both networks have been trained many times and its efficiency tested at the end of each learning process, finding that both systems can be 100% efficient in many cases. Taking into account this fact, the mean efficiency over a batch of instances of the same problem has been measured, and this parameter used t decide which network performs better in a statistical sense.

The BM used in the comparison is fully connected, with four output units and a variable number of hidden neurons ranging from zero to two. Learning parameters are once again $\eta = 0.2$, $\alpha = 0.1$, maximum absolute error $|\partial\mathcal{E}/\partial w| = 0.05$ and maximum absolute initial random value for weights $|w_0| = 1.0$. On the other hand, the topology of the perceptron employed has been optimized to get best results. The experiment has been repeated using a number of hidden units spanning the range from 5 to 2500, using both lineal and hyperbolic tangent transfer functions, and a momentum α between 0.0 and 0.2 with an adaptive η learning rate. Results on the performance are presented in Table 2.

Table 2. Decimation method against perceptron

γ	Mean eff. for BM	Mean eff. for perceptron
0.10	97.25	60.38
0.15	96.87	68.77
0.20	94.58	83.49
0.25	94.05	88.55
0.30	86.90	84.87

It can be seen from the table that the Boltzmann Machine performs slightly *better* than the perceptron. This can be understood when the problem is carefully analyzed, as it has an original discrete nature. Since the Boltzmann Machine is a binary neural network and the perceptron is a continuous one, the BM is

better suited to solve the problem. However, the perceptron is a widely used multi purpose network, and it can perform very well on problems where other continuous models fail. In any case and although the Boltzmann Machine does a better job, the perceptron still provides solutions that are more than satisfactory. Still when High Order Decimation is employed, the Boltzmann Machine not only outperforms the perceptron but also gets the solution in a similar period of time.

6 Summary and Conclusions

In this work a generalization of the standard decimation method in the learning of Boltzmann Machines is presented and discussed, showing that the inclusion of high order weights can simplify the learning process. Comparisons with the standard BM and an optimized perceptron show that the procedure is not only easy to implement but also that its performance in terms of efficiency and computational cost is better. In this way, high Order Decimation may be the key to bring Boltzmann Machines to a level where they become effective in the resolution of practical problems.

References

1. Saul, L., Jordan, M.I.: Learning in Boltzmann Trees. Neural Computation, Vol. 6., Num. 6 (1994) 1174–1184
2. Rüger, S.M.: Decimatable Boltzmann Machines for Diagnosis: Efficient Learning and Inference. World Congress on Scientific Computation, Modeling and Applied Mathematics. Vol. 4: Artificial Intelligence and Computer Science (1997) 319–324
3. Ackley, D.H., Hinton, G.E., and Sejnowski, T.J.: A learning algorithm for Boltzmann machines. Cognitive Science, Vol. 9. (1985) 147–169
4. Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. USA 79 (1982) 2554–2558
5. Hertz, J., Krogh, A., Palmer, R.G.: Introduction to the theory of neural computation. Addison-Wesley Publishing Company, Santa Fe Institute (1991)
6. Kirkpatrick, S., Gelatt Jr, C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science, Vol. 220 (1983) 671–680
7. Kullback, S.: Information theory and statistics. New York: Willey, 2nd ed. (1959)
8. Sejnowski, T.J.: High-Order Boltzmann Machines. AIP Conference Proceedings 151 on Neural Networks for Computing, Snowbird, Utah, E.U.A. (1987) 398–403
9. Albizuri, F.X., d'Anjou, A., Graña, M., Torrealdea, F.J., Hernández, M.C.: The High-Order Boltzmann Machine: Learned distribution and Topology, IEEE Transactions on Neural Networks, Vol. 6, Num.: 3 (1995) 767–770
10. Graña, M., d'Anjou, A., Albizuri, F.X., Hernández, M, Torrealdea, F.J., de la Hera, A., González, A.I.: Experiments of Fast Learning with High Order Boltzmann Machines. Applied Intelligence, Vol. 7, Num. 4 (1997) 287–303
11. Itzykson, C., Drouffe, J.: Statistical field theory. Cambridge University Press (1991)
12. Álvarez, V., Armario, J.A., Frau, M.D., Real, P.: Matrices cocíclicas de Hadamard sobre productos semidirectos. III Jornadas de Matemàtica Discreta y Algorítmica 3JMDA, Sevilla (2002) 155–158
13. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Second Ed., Wiley-Interscience Publication, John Wiley & Sons, INC. (2001)

Bounded Weights Trajectories for Neural Network State Estimation

I. Chairez [1], A. Poznyak [1] and T. Poznyak [2]

[1] Department of Automatic Control, CINVESTAV-IPN. Av. Instituto Politécnico Nacional, Col. San Pedro Zacatenco, C. P. 07360, Mexico D.F., México, Tel: +(525) 55 5061 37 41,Fax: +(525) 55 747 70 89. e-mail: apoznyak@ctrl.cinvestav.mx

[2] Superior School of Chemical Engineering and Extractive Industries (ESIQIE-IPN), Edif. 7, UPALM, C.P. 07738, México, D.F, México, e-mail: tpoznyak@ipn.mx

(Paper received on July 12, 2006, accepted on September 25, 2006)

Abstract. The most important fact on differential neural networks (DNN) dynamics is related to the weights time evolution. This is a consequence for the complex non-linear structure describing the weights matrix differential equations associated with the adaptive capability for this kind of neural networks. However, there is no any analytical demonstration about the weights stability when the DNN approach is considered. In fact, this is the main inconvenient to design real applications for differential neural network observers (DNNO), especially to propose adaptive control functions for uncertain non-linear systems. This paper deals with the stability proof for the weights dynamics using an adaptive procedure to adjust the weights ODE. Three different examples (two of them were realized by numerical simulations and the last one was carried out using real bio-filtering process data) demonstrated the suggested approach performance.

1. Introduction

Artificial Intelligence (AI) is a vast and complicated subject area. Like many others, it is dogged by terminology and unexplained mathematics. At a base level, AI concerns some numbers that are changed by algorithms over time to achieve a certain goal. Many paper and books have been written on intelligent control using neural networks (NN) [1], [2]. NN's and their universal approximation property and learning capability [3] have proven to be a powerful tool to control complex dynamically non-linear systems with parameter uncertainties. Exploiting the fact of being universal approximates, it may straightforwardly substitute unknown system uncertainties by Neural Networks schemes, which is defined by a specific mathematical model (continuous, discrete, etc.) but contains a number of unknown parameters ("weights") to be adjusted. Depending on the suggested neural network model, its free parameters could appear as a linear or non-linear element in the NN description, and they may be modified using differential or difference equations [4], [5]. This kind of adjustment algorithm, transforms the original problem into a nonlinear robust adaptive feedback one. The differential neural network (DNN) approach, exploiting the properties of the applied NN, permits to avoid many problems related to global extreme search (as is usual when backpropagtion NN are applied), converting the learning process to an

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 13-24

adequate feedback design. [5]. If the mathematical model of a considered process is incomplete or partially known, the DNN-approach provides an effective instrument to attack a wide spectrum of problems such as identification, state estimation, trajectories tracking, etc [3].

Lyapunov's stability theory has been used within the neural networks for control literature since fifty years [5]. [6]. This is the main tool to guarantee close-loop system performance. Even though there is a general trend to enlarge the nonlinear systems for which the aforementioned works can be applied. On this field results on stability, convergence to arbitrarily small sets and robustness to modelling imperfections and external perturbations of the closed-loop system, have been established, in order to provide a larger class of non-linear systems that can be treat by this technique. The problem of learning in automatic control systems has been studied in the past, especially during 60s. At that time, learning was analyzed as the estimation or approximation of the unknown quantities in a function, in similar way to other terms as adaptation and self-organizing. Later on, learning theory has become a research discipline in the context of machine learning, and more recently, in computational or statistical learning using stochastic principles [7]. In neural modelling, learning is a path in the space of control parameters driving the system toward phase transition and bifurcations [8]. Though statistic learning theory could provide efficient learning algorithms for a wide variety of problems: in the robust analysis and synthesis of control systems, e.g., [9]. However, the same kind of learning process is complex to understand and difficult to be applied in practical state estimation and control systems, which are mostly dynamic and deterministic by nature.

The main contribution of this paper deals with the justification of the boundedness property for the weights trajectories involved in the suggested DNN state estimator description. By Lyapunov technique, the stable behaviour for the matrix differential learning law (which is also derived along this study) is analyzed. Besides, the adjustable parameters involved in the weights dynamics are suggested to be designed like sigmoid function or the, so-called, inverse one with a predefined rate convergence. Three simple examples demonstrated the workability of the approach provided in this paper.

2. Differential Neural Observer

The class of systems to be treated during this work is described by the vector nonlinear differential equation

$$\dot{x}_t = f(x_t, u_t, t) + \xi_{1,t}, \quad y_t = Cx_t + \xi_{2,t} \tag{1}$$

where $x_t \in \Re^n$ is the system state, $y_t \in \Re^p$ is the system output $(p \le n)$, $u_t \in \Re^m$ is the bounded control action $(m \le n)$ belonging to the following admissible set $U^{adm} := \left\{ u : \|u\|^2_{\Lambda_u} = u^T \Lambda_u u \le v_0 < \infty \right\}$ $\Lambda_u = \Lambda_u^T > 0$. The output matrix $C \in \Re^{p \times n}$ is assumed to be a-priory known. The nominal closed-loop

dynamics is quadratically stable for a fixed control $u_t^* \in U^{adm}$, that is, there exists a Lyapunov function $\bar{V}_t$ such that :

$$\begin{aligned} &\frac{\partial \bar{V}_t}{\partial x} f(x_t, u_t^*, t) \le -\lambda_1 \|x_t\|^2 < 0 \\ &\left\| \frac{\partial \bar{V}_t}{\partial x} \right\| \le \lambda_2 \|x_t\|^2 < \infty, \quad \lambda_1, \lambda_2 > 0 \end{aligned} \tag{2}$$

The vectors $\xi_{1,t}$ and $\xi_{2,t}$ represent the state and output deterministic bounded (unmeasurable) disturbances, i.e., $\|\xi_{j,t}\|^2_{\Lambda_{\xi_j}} \le \Upsilon_j$, $\Lambda_{\xi_j} > 0$, $j=1,2$, and do not violate the existence of the solution to ODE (1). The nominal output system (without external perturbations $\xi_{2,t} = 0$) is uniformly observable [10], that is, the following (observability) matrix is no singular for any $t \ge 0$:

$$\tilde{O} := \nabla_x \left[C^T, \left[L_f(Cx_t)\right]^T, \left[L_f^2(Cx_t)\right]^T, \ldots, \left[L_f^{n-1}(Cx_t)\right]^T \right]^T \tag{3}$$

Here $L_f(\cdot)$ is the Lie derivative operator [10]. To ensure the uniqueness and the existence of the non-linear dynamics, it is supposed the class of non-linear functions in (1) satisfies the Lipschitz condition (uniform on t) on two first arguments, that is,

$$\begin{aligned} &\|f(x,u,t) - f(y,v,t)\| \le L_1 \|x - y\| + L_2 \|u - v\| \\ &\|f(0,0,t)\|^2 \le C_1; \;\; x, y \in \Re^n; \;\; u, v \in \Re^m; \;\; 0 \le L_1, L_2 < \infty \end{aligned} \tag{4}$$

The last assumption automatically implies the following cone-property

$$\|f(x_t, u_t, t)\|^2 \le C_1 + C_2 \|x_t\|^2 \tag{5}$$

Which is valid for any x, u and t. Notice that (1) always could be represented as

$$\begin{aligned} &\dot{x}_t = f_0(x_t, u_t, t \mid \Theta) + \tilde{f}_t + \xi_{1,t} \\ &\tilde{f}_t := f(x_t, u_t, t) - f_0(x, u, t \mid \Theta) \end{aligned} \tag{6}$$

where $f_0(x,u,t \mid \Theta)$ is treated as a possible "*nominal dynamics*" which can be selected according to a designer desires and $\tilde{f}_t$ is a vector called the "*no modelled dynamics*". Here the parameters Θ are subjected to adjustment in order to obtain the complete matching between the nominal and the non-linear dynamics. In view of (6) and the corresponding boundedness property, the following upper bound for the no

modelled dynamics $\widetilde{f}_t$ takes place:

$$\left\|\widetilde{f}_t\right\|^2_{\Lambda_f} \le \widetilde{f}_0 + \widetilde{f}_1\|x_t\|^2_{\Lambda^1_{\widetilde{f}}},\ \Lambda_f, \Lambda^1_{\widetilde{f}} > 0 \tag{7}$$

According to DNN approach [5], we will define the nominal dynamics as

$$\begin{gathered} f_0(x,u,t \mid \Theta) = A^0 x + W_1^0 \sigma(x) + W_2^0 \varphi(x) u \\ \Theta := \left[W_1^0, W_2^0\right],\ A^0 \in \Re^{n\times n}, W_1^0, W_2^0, \in \Re^{n\times n}, \sigma \in \Re^{l\times 1}, \varphi \in \Re^{n\times m} \end{gathered} \tag{8}$$

The activation vector-functions $\sigma_t(\cdot)$ and $\varphi(\cdot)$ are usually constructed with sigmoid function components

$$\begin{gathered} \sigma_j(x) := a_j\left(1 + b_j \exp\left(-\sum_{j=1}^{n} c_j x_j\right)\right)^{-1} \\ \varphi_{kl}(x) := a_{kl}\left(1 + b_{kl} \exp\left(-\sum_{j=1}^{n} c_{kl} x_l\right)\right)^{-1} \\ j = \overline{1,n},\quad k = \overline{1,n},\quad l = \overline{1,m} \end{gathered} \tag{9}$$

It is easy to prove that each component in the activation functions satisfies the following sector conditions

$$\|\sigma(x) - \sigma(x')\|^2_{\Lambda_\sigma} \le l_\sigma \|x - x'\|^2_{\Lambda_\sigma},\quad \|(\varphi(x) - \varphi(x'))u_t\|^2_{\Lambda_\varphi} \le l_\varphi v_0 \|x - x'\|^2_{\Lambda_\varphi} \tag{10}$$

2.1. DNN structure containing relay correction term

Lets introduce the adaptive state estimator based on DNN [11] as follows:

$$\begin{gathered} \frac{d}{dt}\hat{x}_t = A^{(0)}\hat{x}_t + W_{1,t}\sigma(\hat{x}_t) + W_{2,t}\varphi(\hat{x})u_t + \\ K_1[y_t - C\hat{x}_t] + K_2 \frac{y_t - C\hat{x}_t}{\|y_t - C\hat{x}_t\|},\quad \hat{y}_t = C\hat{x}_t \end{gathered} \tag{11}$$

So, when $\hat{y}_t = C\hat{x}_t$, ODE (11) should be attended as a differential inclusion [12]. Here the weights matrices ($W_{1,t}, i = 1,2$) supply the adaptive behaviour to this class of observers. The non-linear weight *updating* (learning) law is described by the following differential equations

$$\begin{aligned} \dot{W}_{i,t} &= -k_{i,t}\Xi_t + 2^{-1}k_{i,t}^{-1}\dot{k}_{i,t}\hat{W}_{i,t}, \quad i=1,2, \quad j=1,2 \\ \Xi_t &:= P_1 N_\delta \Omega_{i,t}\Psi_i^T, \qquad \Psi_1 = \sigma(\hat{x}_t), \quad \Psi_2 = \varphi(\hat{x})u_t \\ \Omega_{i,t} &:= \left(\delta^2\Lambda_j + C^T\Lambda_{\xi_2}C\right)N_\delta P_1\hat{W}_{i,t}\Psi_t + C^T e_t, \end{aligned} \tag{12}$$

Here, the matrix P is the positive definite solution for the Riccati equation

$$\begin{aligned} P\widetilde{A}^{(0)*} + \left(\widetilde{A}^{(0)*}\right)^T P + PRP + Q = 0 \\ \overline{A} := A^{(0)*} - K_1 C \\ Q := \Lambda_1^{-1} + \Lambda_2^{-1} + \Lambda_\sigma + v_0\Lambda_\varphi + Q_0 \\ R := W_1^0\Lambda_\sigma^{-1}\left[W_1^0\right]^T + W_2^0\Lambda_\varphi^{-1}\left[W_2^0\right]^T + \Lambda_{\bar{f}} + \Lambda_{\xi_1} + K_1\Lambda_{\xi_2}K_1^T + K_2\Lambda K_2^T \end{aligned} \tag{13}$$

The problem analyzed here could be stated as follows: *Under the conditions A1-A4 for any admissible control strategy* u_t belonging to U^{adm}, *selecting matrices* $A^{(0)}$, K_1, K_2 *and designing the update law (12) (including the selection of* $W_i^{(0)}, i=1,2$*) in such a way that the upper bound for the averaged estimation error* β *defined as*

$$\beta := \limsup_{t\to\infty}\frac{1}{t+\varepsilon}\int_{s=0}^{t}\left\|\hat{x}_s - x_s\right\|_{Q_0}^2 ds, \quad \varepsilon, Q_0 > 0 \tag{14}$$

would be, as less as possible.

Theorem 1. *If there exist positive definite matrices* $\Lambda_{\bar{f}}$, Λ_{ξ_1}, Λ_{ξ_2}, Λ_σ, Λ_D, Λ_u, Λ_1, Q_0 *and positive constants* δ, k, v_1 *such that the matrix Riccati equation (13) has positive definite solution, then the DNN observer (11) with any matrix* K_1 *guarantying that the close-loop matrix* $\widetilde{A}^{(0)*}$ *is stable, that is,* $\widetilde{A}^{(0)*} := \left(A^{(0)*} + K_1 C\right)$ is Hurwitz *and* $K_2 = \lambda P_1^{-1}C^T, \lambda > 0$ *supplied by the learning law (12), provides the following upper bound for the state estimation process:*

$$\begin{aligned} \overline{\lim_{t\to\infty}}\frac{1}{T}\int_{t=0}^{T}\left\|\Delta_t\right\|_P^2 dt \le \rho_Q/\alpha_Q \\ \rho_Q := \bar{f}_0 + v_0 + \Upsilon_1 + 3\Upsilon_2 + 8\lambda\sqrt{n\Upsilon_2}, \quad \alpha_Q := \lambda_{\min}\left(P_1^{-1/2}Q_0P_1^{-1/2}\right) > 0 \end{aligned} \tag{15}$$

The proof of this theorem could be done using the ideas given in [5].

2.2. Stable trajectories for the weights dynamics

Once the upper bound for the estimation process has been derived, it is possible

(independently) to consider the stability analysis on the learning laws obtained during the observer development. The study was conducted suggesting a new Lyapunov-like function V_1 for the weight $\widetilde{W}_{1,l}$ (and the corresponding V_2 to $\widetilde{W}_{2,l}$) and applying the conventional procedure to determine the bounded behaviour for the weights dynamics. This result could be abstracted in the following:

Theorem 2. *The weights time trajectories are stable (in Lyapunov sense).*

Proof. Let considering the dynamics for the weight matrix $\widetilde{W}_{1,l}$ and the following (suggested) Lyapunov function V_1.

$$V_1 := \tfrac{1}{2} tr\left\{\widetilde{W}_{1,l}^T \widetilde{W}_{1,l}\right\} + \tfrac{c}{4}\left[k_{1,l} - k_{1,\min}\right]_+^2, \quad [z_l]_+ := \begin{cases} z & z \geq 0 \\ 0 & z < 0 \end{cases} \tag{16}$$

Which immediately implies (by direct differentiation)

$$\begin{aligned} \dot{V}_1 &:= tr\left\{\widetilde{W}_{1,l}^T\left(-k_{1,l}\Xi_1 + 2^{-1}k_{1,l}^{-1}\dot{k}_{1,l}\widetilde{W}_{1,l}\right)\right\} + 2^{-1}c\dot{k}_{1,l}\left[k_{1,l} - k_{1,\min}\right]_+ \\ &\leq k_{1,l}\left|tr\left\{\widetilde{W}_{1,l}^T\Xi_1\right\}\right| + 2^{-1}\dot{k}_{1,l}\left(\frac{1}{k_{1,l}} tr\left\{\widetilde{W}_{1,l}^T\widetilde{W}_{1,l}\right\} + c\left[k_{1,l} - k_{1,\min}\right]_+\right) \end{aligned} \tag{17}$$

If $\dot{V}_1$ is desired to be non positive, the free parameter $\dot{k}_{1,l}$ could be adjusted to obtain the asymptotic behaviour for the DNN weights dynamics in the following manner

$$\dot{k}_{1,l} \leq -\frac{2k_{1,l}^2\left|tr\left\{\widetilde{W}_{1,l}^T\Xi_1\right\}\right|}{tr\left\{\widetilde{W}_{1,l}^T\widetilde{W}_{1,l}\right\} + ck_{1,l}\left[k_{1,l} - k_{1,\min}\right]_+} \tag{18}$$

The same approach could be apply to $\widetilde{W}_{2,l}$ structure, resulting the corresponding upper value for

$$\dot{k}_{2,l} \leq -\frac{2k_{2,l}^2\left|tr\left\{\widetilde{W}_{2,l}^T\Xi_2\right\}\right|}{tr\left\{\widetilde{W}_{2,l}^T\widetilde{W}_{2,l}\right\} + ck_{2,l}\left[k_{2,l} - k_{2,\min}\right]_+} \tag{19}$$

These two conditions avoid the finite time escape for the weights dynamics. This tool could be used specially when there is any possible feedback control using the Weights Dynamics like adaptive parameter.

2.3. Learning Laws Examples

In this subsection, two simple examples for adaptive learning laws are introduced using

the next definition

$$s\left(\tilde{W}_{j,t}, e_t\right) := 2k_{j,t}^2 \left| tr\left\{\tilde{W}_{j,t}^T \Xi_j\right\} \right| \left[tr\left\{\tilde{W}_{j,t}^T \tilde{W}_{j,t}\right\} + ck_{j,t}\left[k_{1,t} - k_{1,\min}\right]_+ \right]^{-1} \quad (20)$$

a) Sigmoid learning law. The first example uses the sigmoid representation:

$$k_{j,t} := k_{0,j}\left(1 + a\left(\tilde{W}_{j,t}^T, e_t\right)\exp\left(b_j t\right)\right)^{-1} + k_{\min,j}, \quad k_{\min,j} > 0 \quad (21)$$

The corresponding time history $\dot{k}_{j,t}$ could be easily derived

$$\dot{k}_{j,t} := -k_{j,t} a\left(\tilde{W}_{j,t}^T, e_t\right) b_j \exp\left(b_j t\right)\left(1 + a\left(\tilde{W}_{j,t}^T, e_t\right)\exp\left(b_j t\right)\right)^{-1} \quad (22)$$

which itself implies

$$a\left(\tilde{W}_{j,t}^T, e_t\right)\exp\left(b_j t\right)\left(k_{j,t} b_j - s\left(\tilde{W}_{j,t}^T, e_t\right)\right) > s\left(\tilde{W}_{j,t}^T, e_t\right) \quad (23)$$

Last inequality is fulfilled if and only if the weight dependent parameter $a\left(\tilde{W}_{j,t}^T, e_t\right)$ is selected in such a way

$$a\left(\tilde{W}_{j,t}^T, e_t\right) > s\left(\tilde{W}_{j,t}^T, e_t\right)\exp\left(-b_j t\right)\Psi_j^{-1}, \quad \Psi_j := k_{j,t} b_j - s\left(\tilde{W}_{j,t}^T, e_t\right) \quad (24)$$

b) Inverse learning law with predefined rate convergence. The second adaptive learning law is described by the inverse learning law with predefined rate convergence:

$$k_{1,t} := k_0\left(1 + a\left(\tilde{W}_{1,t}^T, e_t\right)t^k\right)^{-1} + k_{\min}, \quad k_{\min} > 0, \quad a\left(\tilde{W}_{2,t}^T, e_t\right) > 0 \quad (25)$$

It is easy to proof the time derivative for this function is described like

$$\dot{k}_{1,t} = k_0 k a\left(\tilde{W}_{1,t}^T, e_t\right)t^{k-1}\left[1 + a\left(\tilde{W}_{1,t}^T, e_t\right)t^k\right]^{-2}, \quad k > 1, \quad k \in \aleph \quad (26)$$

The free parameter $a\left(\tilde{W}_{1,t}^T, e_t\right)$ should fulfil the following restriction

$$a\left(\tilde{W}_{1,t}^T, e_t\right) \geq \frac{k_0 - 2s\left(\tilde{W}_{1,t}^T, e_t\right)t^k \pm \sqrt{k_0^2 - 4k_0 s\left(\tilde{W}_{1,t}^T, e_t\right)t^k}}{2s\left(\tilde{W}_{1,t}^T, e_t\right)t^k} > 0 \quad (27)$$

It is important to notice that the learning laws must be adjusted on-line, i.e. each change on the weights values implies the corresponding change in the parameter $a(\cdot)$, so the adaptive process is carried out in two steps: the first one with the learning law application and second, the modification in the learning law rates $k_{j,t}$.

3. Numerical Examples

Three different numerical examples are analyzed using the suggested method: a benchmark problem represented by a linear system, a chemical model dealing with the ozonation process to treat water and finally the data corresponding to a real biological process called biofilter method.

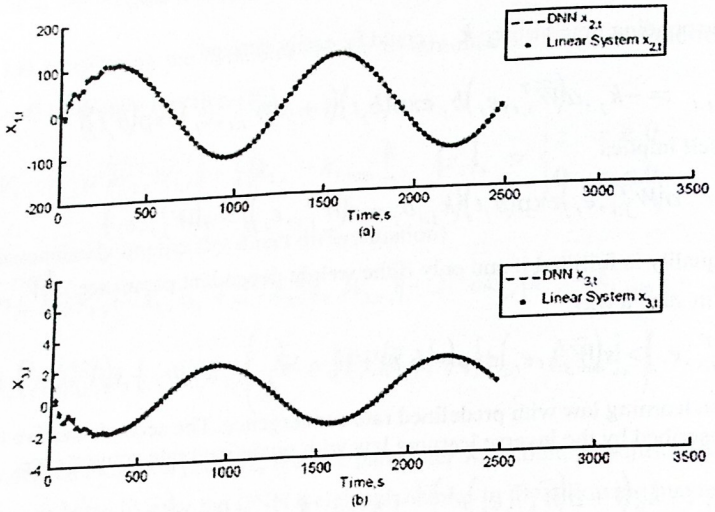


Figure 1. State estimation procedure for the linear system. a) State one non-parametric observation procedure and b) State three non parametric estimation procedures.

a) Benchmark problem: Linear System

Let consider the linear system given by:

$$A := \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -4 & -5 & -2 \end{bmatrix}, \quad B := \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{28}$$

$$C^T := [1 \quad 0 \quad 0], \; u(t) := 3\sin(t) + WN(0.1)$$

Here $WN(0.1)$ is the standard pseudo-white noise signal generated by Matlab-simulink. This example is presented like a benchmark problem in order to prove the workability of this "adaptive" identification approach for simple systems. The estimation process was carried out considering the adaptive learning law ensuring the bounded property for the weights involved in the DNN structure. The DNNO

application on the linear system () demonstrated the complete convergence between the state given by the linear system (x_1 is assumed to be completely accessed by any physical mean) and the state given by the suggested approach (Fig. 1).

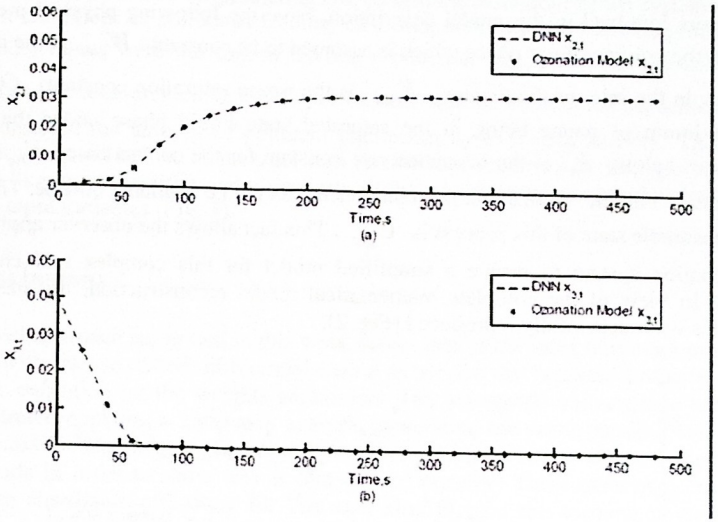


Figure 2. State estimation procedure for the ozonation system. a) Reconstructed state Q_1 (dissolved oxygen) given by DNN observer and b) decomposition dynamics for the contaminant C_1' derived by non-parametric estimation procedure.

b) Ozonation reaction: mathematical model approach

Ozone is capable to oxidize a variety of organic materials in aqueous solution. The oxidation process by ozone involves the phenomenon of mass transfer with simultaneous chemical reaction. Based on the results obtained in [11], where the simple ozonation with i-component model mixture at the pH 7 has been treated. This process can be described using the following system of ODE:

$$\begin{aligned} \dot{C}'_{gas} &= \frac{1}{V_{gas}}\left[W_{gas}\left(C^{in}_{gas} - C'_{gas}\right) - K_{sat}\left(Q_{\max} - Q'\right) + k_1 C'_1 Q'\right] \\ \dot{Q}' &= K_{sat}\left(Q_{\max} - Q'\right) - k_1 C'_1 Q', \quad \dot{C}'_1 = -\frac{k_1 C'_1}{V_{liq}} Q' \end{aligned} \tag{29}$$

where C'_{gas} is the ozone concentration in the output of the reactor (this is ozone which doesn't react with organic compounds dissolved in the solvent), Q_l is ozone dissolved in a liquid phase. C'_1 is the organic compound concentration at time t. The parameters involved in the model description, have the following physical meaning: V_{gas} is the volume of gas phase which is assumed to be constant, W_{gas} is the oxygen gas flow in the inlet of the reactor, K_{sat} is the ozone saturation constant, $Q_{\max}$ is the maximum of ozone being in the saturated state liquid phase under the given conditions below, k_1 is the ozonation rate constant for the contaminate, V_{liq} is the liquid phase volume. Notice that this component can not be available on-line. The only one measurable state of this process is C'_{gas}. This fact allows the observer application on ozonation system to derive a simplified model for this complex real chemical system. In view of the complete mathematical model reconstruction, the ozonation dynamics was successfully reproduced (Fig. 2).

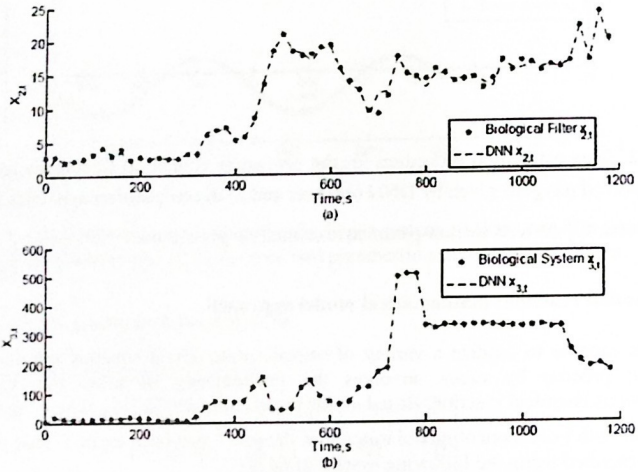


Figure 3. Observation procedure for bio-filtering method. a) Reconstructed state $(x_{2,t})$ (carbone dioxide) given by DNNO and b) pressure drop $(x_{3,t})$ derived by non parametric estimation procedure

c) Real Data Adjustment: Biological Filter problem

The popularity of the biofiltration has been increasing due to the high amount of

pollutants produced by the industrial and human activity and the advantages offered by this technology. A lot of studies were conducted to enhance the performance of biofiltration systems. Consequently, there is a necessity to develop methodologies enabling to predict and determine the reactor performance, not only to design more efficient reactors but also for predicting and control the behaviour of the systems under different conditions of pollutants feeding, air flows, humidity and biomass production. The model inputs were the carbon dioxide production $(x_{2,t})$ and the pressure drop $(x_{3,t})$ (on line measurements), the output variable was the elimination capacities (CE), which describe the amount of pollutant eliminated by microbial activity. The DNN observer was used here to reconstruct both on line measurements, so this approach can be used like input identifier and to reconstruct the relationship between the plant states and the input variables (Fig. 3).

4. Conclusions

The novel approach suggested in this work solves one of the most important problems related with the, so-called, differential neural networks: the bounded property of the dynamic evolution for the weights parameters. The asymptotic convergence has been demonstrated applying a Lyapunov analysis, generating the corresponding conditions for the possible learning rate function. Using these conditions, two different suggestion were made in order to show why is this method feasible. Three numerical examples show the simulation efficiency for this new kind to treat the learning procedure in DNN.

References

[1] W. T. Miller, R. S. Sutton, and P. J. Werbos. Neural Networks for Control. MIT Press, 1990.

[2] K. Narendra and K. Parthasarathy. "Identification and control of dynamical systems using neural networks," IEEE Transactions on Neural Networks, vol. 1, pp. 4-27, January, 1990.

[3] F. L. Lewis, A. Yesildirek, and K. Liu, "Multilayer neural-net robot controller with guaranteed tracking performance," IEEE Trans. Neural Netw., vol. 7, no. 2, pp. 1-11, 1996.

[4] S. Haykin. Neural Networks, a comprehensive Foundation. New York: IEEE Press, 1994.

[5] A. Poznyak, E. Sanchez, and W. Yu. Differential Neural Networks for Robust Nonlinear Control (Identification, State Estimation and Trajectory Tracking. World Scietific, 2001.

[6] G. Rovithakis and M. Christodoulou, "Adaptive control of unknown plants using dynamical neural networks," IEEE Trans. Syst., Man and Cyben., vol. 24, pp. 400-412, 1994.

[7] V. N. Vapnik, The Nature of Statistical Learning Theory. New-York: Springer-Verlag, 2nd ed., 2000.
[8] J. Tani, "Model-based learning for mobile robot navigation from the dynamical system perspective," IEEE Trans. System, Man and Cybernetics part B, vol. 26, no. 3, pp. 421-436, 1996.
[9] M. Vidyasagar, "Randomized algorithms for robust controller synthesis using statistical learning theory," Automatica, vol. 37, no. 10, pp. 1515-1528, 2001.
[10] A. Krener and A. Isidori, "Linealization by output injection and nonlinear observers," Systems and Control Letters, vol. 3, pp. 47-52, 1983.
[11] T. Poznyak and A. Chairez, I. Poznyak, "Application of a neural observer to phenols ozonation in water: Simulation and kinetic parameters identification," Water Research, vol. 39, pp. 2611-2620, 2005.
[12] A. Poznyak, Sliding Modes: From Principles to Implementation, ch. Chapter 3: Deterministic Output Noise Effects in Sliding Mode Observation, pp. 123-146. IEE Press, 2001.

Hepatitis C Viral Load Control Applying Hamilton Jacobi And Sliding Mode Techniques

R. F. Miranda, J. I. Chairez and A. Cabrera
Av. Acueducto de Guadalupe s/n Col. Ticoman Barrio La Laguna, CP. 07427
México, D.F.
(Paper received on August 11, 2006, accepted on September 26, 2006)

Abstract. Hepatitis C is one of the illness which has affected a great deal of people all over the world, it is suffered approximately by the world population's three percent and it has become an important social problem that has shown to be in continuous growing. For that strong reason it is presented a control description based on Hamilton-Jacobi Technique (HJT) and Sliding Mode Approach (SMA) working jointly with a nonlinear observer (NO), carried out with Differential Neural Networks (DNN) and Sliding Mode Approach, the one which has proven to be useful in real medical procedures to expense of the mathematical model that describes the Hepatitis C Virus (HCV) dynamics is similar to the real Interferon Alpha 2b (IFN $\alpha - 2b$) patients response. This Hamilton Jacobi and Sliding Mode Control (HJSMC) gives us the exact dosage that ought to be given to a patient with the purpose of maintaining controlled, in very low levels, the viral load.

1. Introduction

Mathematics, control techniques and computational intelligence have been used in the study of a great deal of process showing to be quite fruitful when having good results, specially in biological sciences because computational and mathematical models are helping biological scientist to know many aspects of the complex realm of living matter [2], [3]. In particular cases, these mathematical principles have been used to understand the life cycle of chronic viral infections development [4], [5] and the behavior under several treatments of Hepatitis C infection [2], [6]. The simulation results demonstrate how the present techniques are allowed to create software to solve emerging problems of therapy optimization, however, many of these models do not consider all possible systems variations and they strongly depend on the input function. Hepatitis C Virus (HCV) has infected about 170 million of people around the world [7], and this sickness could provoke serious injuries in hepatic tissue like cirrhosis, which has a slow and progressive development and the ways of contracting this illness are: sharing intravenous devices, sexual contact, using the parenteral pathway, etc. [8]. Despite the important scientific and technological advances in medicine, there in no cure or vaccine the one that control the virus or keep it contained until now. The main method proven until now uses some pharmacological products as Lamivudine, Adefovir, Ribavirine and the Interferon alpha presented as Interferon alpha - 2a and Interferon alpha 2b (IFN $\alpha - 2b$) whose many studies has demonstrated that is the best option to treat Hepatitis C [9]. There are some processes that have gain scientific groups' attention like chemical and biological ones (i.e. hepatitis C virus dynamics) because exists several nonlinearities in they structure and almost always they are

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 25-35

difficult to model since the resulting models are often heavily dependent on the number of states and the parameter set. in addition to that they are very expensive in computational time. Nonlinear black box models may overcome this problem because this method does not use the internal process of the model; they just need the input-output data pair. Artificial Neural Networks (ANN) are computational tools that have obtained considerable attention in biological studies as good as academic research or industrial applications. Their universal approximation abilities and the access to a wide range of software tools qualify the ANN for the building of nonlinear dynamics black-box models which can be applied as prediction models or state estimator devices. Two types of ANN are known: static, they are those that use the so-called back-propagation technique (for example) [10], [11] and dynamic neural networks [12], [13]. The second approach, exploiting the feedback properties of the applied DNN, allows avoiding many problems related to global extremum search converting the updating (training) process to an appropriate feedback design. As it has already been seen in [14] to carry out a control with DNN trajectory tracking, it is necessary to identify the estimated states by a DNN identifier to guarantee the convergence of the error, what implies extra computational spending. With Hamilton Jacobi's Algorithm for trajectory tracking it is possible to simplify this spending since this control algorithm uses the estimated states of a DNN. The system's behavior of prospective control is to displace the states such that it is possible to make a track of a sign generated by some reference model. The Sliding Mode Approach (SMA) allows powerful advantages than other identification and control techniques like good transient behavior, global exponential stability with small estimate error, capability to reject no modeled disturbances, insensitivity to plant nonlinearity or parameter variations and remarkable stability and performance robustness [15], [16]. Such in [17] the SMA approach is used to obtain the algebraic (non differential) weight-learning procedure for on-line identification of a nonlinear plant with completely available states. DNN observers containing sign-term are also considered in [17] but in that approach, the weight adjustment is governed by a "standard" Ordinary Differential Equation (ODE) with time varying parameters. No relay terms are used within the learning procedure.

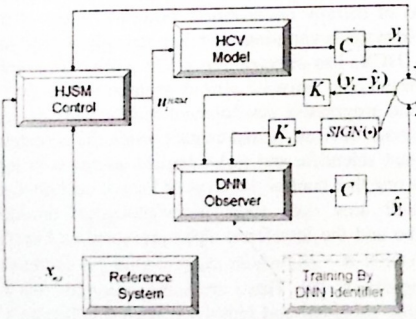


Fig. 1. Block diagram of the complex process.

2. Methodology

The corresponding methodology was carried out in three procedures (figure 1): the first one is the training process with a DNN identifier, then the estimate procedure using a DNN observer and finally the proper control via Hamilton Jacobi and SMA.

2.1 Mathematical Model of the Hepatitis C Viral Load Dynamics

The mathematical model that describes the dynamics of the HCV and their changes when IFN $\alpha - 2b$ is applied into patient is:

$$\dot{T}_t = s - dT_t - (1-\eta)\beta V_t T_t; \quad \dot{I}_t = (1-\eta)\beta V_t T_t - \delta I_t; \quad \dot{V}_t = (1-u_t\varepsilon)pI_t - cV_t \tag{1}$$

where $\dot{T}_t$ [mIU / mL] describes changes in the number of target cells $\dot{I}_t$ [mIU / mL] represents variations in the number of productively infected cells and $\dot{V}_t$ [mIU / mL] shows how viral load changes in time when IFN $\alpha - 2b$ is applied as an input function u_t. The target cell's production is given by the constant rate s $(1000[mL\,dia]^{-1})$ and these cells die with a constant rate d $(0.014[mL\,dia]^{-1})$, η (0.001) is the possible effect of the IFN $\alpha - 2b$ and the immunological system to contain the *novo* infection given by β $(3\times10^{-7}[mL\,dia]^{-1})$ which is the constant rate associated with novo infection entrance, once the target cells become productively infected cells, they die by the constant rate δ $(0.14[mL\,dia]^{-1})$, ε (0.8) is the possible effect of the IFN to break viral production, p $(100[mL\,dia]^{-1})$ is the constant rate of new virus reproduction and c $(6.2[mL\,dia]^{-1})$ is the viral clearances constant rate. This analysis searches the major initial effect of IFN $\alpha - 2b$ is to break viral production. This mathematical model plays an important roll since the whole work carried out in this paper supposes that the model resembles each other to the patients' real response and this can be used for the corresponding computer simulations.

HCV model (1) could be generalized, in a mathematical sense, as (2).

$$\dot{x}_t = f(x_t, u_t, t) + \xi_{1,t}; \qquad y_t = Cx_t + \xi_{2,t} \tag{2}$$

where $x_t \in \Re^n$ is the state vector, $y_t \in \Re^p$ is a linear combination of the state elements that it is commonly assumed as the output system, $u_t \in \Re^m$ is the external supply of IFN $\alpha - 2b$, $C \in \Re^{p\times n}$ selected as $C = [0 \;\; 0 \;\; 1]$ since the viral load is the only measurable variable using clinical tests like PCR probe [5]. The vectors $\xi_{1,t}$ and $\xi_{2,t}$ represent the state and output bounded (immeasurable) disturbances, i.e., $\|\xi_{j,t}\|^2_{\Lambda_{\xi_j}} \le \Upsilon_j$, $\Lambda_{\xi_j} > 0$, $j = 1, 2$ that could be associated with the different medical responses of each patient and the experimental errors in the PCR probe [18]. Hereinafter it is supposed that the class of nonlinear functions in (2) satisfies all the restrictions associated with the existence of the differential equation solution. Notice

that the nonlinear system (2) could always be described as: $\dot{x}_t = f_0(x_t, u_t, t\,|\Theta) + \tilde{f}_t$ and $\tilde{f}_t := f(x_t, u_t, t) - f_0(x_t, u_t, t\,|\Theta) + \xi_1$, where $f_0(x_t, u_t, t\,|\Theta)$ is is known as *"nominal dynamics"* which can be selected according to the DNN theoretical results [17] and $\tilde{f}_t$ is a vector called *"no modelled dynamics"* that should be minimized during the common training process. According to DNN approach, the nominal dynamics are assumed as:

$$f_0(x_t, u_t, t \mid \Theta) = A^{(0)}x + W_1^{(0)}\sigma(x) + W_2^{(0)}\varphi(x)u$$

$$\Theta := A^{(0)}, W_1^{(0)}, W_2^{(0)} \in \Re^{n\times n}$$

$$W_1^{(0)} \in \Re^{n\times l}, \sigma(\cdot) \in \Re^{l\times 1}, \quad W_2^{(0)} \in \Re^{n\times l}, \varphi(\cdot) \in \Re^{l\times m}$$

The activation functions $\sigma(\cdot) := [\sigma_i(\cdot)]_{l\times 1}$ and $\varphi(\cdot) := [\varphi_{ij}(\cdot)]_{l\times m}$ are chosen sigmoidal functions i.e. $\sigma_i^{-1}(x) := a_{\sigma i}^{-1}\left(1 + b_{\sigma i}\exp\left(-\sum_{j=1}^{n} c_{\sigma_{ij}} x_j\right)\right)$ and $\varphi_{ij}^{-1}(x) := a_{\varphi ij}^{-1}\left(1 + b_{\varphi ij}\exp\left(-\sum_{j=1}^{n} c_{\varphi_{ij}} x_j\right)\right)$. The admissible control set for u_t is supposed to be described as a state estimate feedback $U^{adm} := \left\{u = u(\hat{x}) : \|u\|_{\Lambda_u}^2 \le v_0 + v_1\|\hat{x}\|_{\Lambda_x}^2\right\}$ where $\hat{x}$ is a estimated state and the matrix $\Lambda_u > 0$.

2.2. Procedure 1. Training Process

Let's consider the dynamics of a patient infected with HCV without previous treatment described by (2) where the entire state vector is known by any clinical method. So, a DNN will be applied to derive the best possible approximation to this nonlinear model just by using the input-output data pair. The mathematical description of this kind of DNN is:

$$\dot{\hat{x}}_t = A\hat{x}_t + W_{1,t}\sigma(\hat{x}) + W_{2,t}\varphi(\hat{x})u_t \tag{3}$$

where $\hat{x}_t \in \Re^n$ is the state vector of the DNN $u_t \in \Re^r$ is a measurable bounded control action, $A \in \Re^{n\times n}$ is a Hurwitz matrix, $W_{1,t} \in \Re^{n\times l}$ is the weight matrix for nonlinear state feedback and $W_{2,t} \in \Re^{n\times m}$ is the input weight matrix. The vector field $\sigma(\hat{x}_t): \Re^n \to \Re^l$ is designed to have elements with a monotonically increasing behavior just like sigmoid functions. The function $\varphi(\,)$ is the transformation from $\Re^r$ to $\Re^{n\times m}$ which is constructed by sigmoid activation functions in each element. The input function $u(\,)$ is assumed to be bounded as $\|u_t\|^2 \le \bar{u}$. The designed identifier requires the next technical fact: There

exists a positive defined matrix Q such that the Ricatti equation (4) has a positive solution $P = P^T > 0$.

$$A^T P + PA + PRP + Q = 0 \tag{4}$$

Theorem 1. Considering the nonlinear system (2) and a model matching neural network (3) whose weights are adjusted by the following matrix differential equations [17]:

$$\begin{aligned} \dot{W}_{1,t} &= -K_1 P \Delta_t \sigma(\hat{x}_t)^T \\ \dot{W}_{2,t} &= -K_2 P \Delta_t \gamma\left(u_t\right)^T \varphi(\hat{x}_t)^T \\ \Delta_t &:= x_t - \hat{x}_t \end{aligned} \tag{5}$$

where K_1 and K_2 are matrices with positive entries, and $P = P^T > 0$ is the solution of the algebraic Ricatti equation given by (4). Then the weights dynamics are bounded: $W_{1,t} \in L_\infty$ and $W_{2,t} \in L_\infty$ and they converge to their best possible values, i.e. $\lim_{t\to\infty} \dot{W}_{1,t} = 0$, $\lim_{t\to\infty} \dot{W}_{2,t} = 0$. Furthermore, it is possible to conclude that the identification process is asymptotically consistent, i.e., $\lim_{t\to\infty} \Delta_t = 0$.

The main element to construct the convergence proof in this scheme is by constructing a Lyapunov function in order to derive the learning laws. This Lyapunov function (6) was selected as:

$$\begin{aligned} V_t &= \Delta_t^T P \Delta_t + tr\left[\tilde{W}_{1,t}^T K_1^{-1} \tilde{W}_{1,t}\right] + tr\left[\tilde{W}_{2,t}^T K_1^{-1} \tilde{W}_{2,t}\right] \\ \tilde{W}_{j,t} &= W_{j,t} - W_{j,0}, \quad j = 1, n \end{aligned} \tag{6}$$

The training process was carried out just one time using the available data for a sick patient. This procedure generates the final values of the weights matrixes and allows the correct selection of the free parameter of the differential neural network as observer.

2.3. Procedure 2. Estimate Process

The estimate process gives us the necessary information required for controlling the HCV. The importance here is to create a system that responds the nearest possible to a real patient, considering different patients response to IFN $\alpha - 2b$ supply The nonlinear system representing the HCV dynamics is described by (2). Let's define a DNN observer as follows:

$$\begin{aligned} \frac{d}{dt}\hat{x}_t &= A^{(0)}\hat{x}_t + W_{1,t}\sigma(\hat{x}_t) + W_{2,t}\varphi(\hat{x}_t)u_t + K_1\left(y_t - C\hat{x}_t\right) + K_2 SIGN\left(y_t - C\hat{x}_t\right) \\ \hat{y}_t &= C\hat{x}_t \end{aligned} \tag{7}$$

This DNN structure is almost the same than the DNN identifier just by adding two new terms: the K_1 multiplied by the output error, which gives the Luenberger type observer and the K_2 multiplied by the sign of the error, which gives the sliding mode structure. The $SIGN(\cdot)$ function (8) is referred to:

$$SIGN(y_f) = \left(sgn(y_{1,f}), \ldots, sgn(y_{n,f})\right) \qquad SIGN(z) = \left(\frac{z_1}{|z_1|}, \ldots, \frac{z_n}{|z_n|}\right)$$
$$\forall x \in \Re^n, x_f \neq 0 \tag{8}$$

Hereinafter the following assumption is supposed to be validated: the matrixes pair (C, A) is observable. This nonlinear estimator is supplied with its own updating (learning) laws given by the nonlinear differential equations (9):

$$\begin{aligned}
\frac{d}{dt}\tilde{W}_{j,f} &= \Phi_{jw}\left(W_{j,f}, \hat{x}_f, y_f, u_f, t \,\middle|\, W_{j,*}^{(0)}\right) \\
\Phi_{jw}(\cdot) &:= -k_{wj} P_1 N_* \left(Ce_f + \frac{\Pi}{2} N_* P_1 \tilde{W}_{k,f} x_f\right) x_f^T \\
x_1 &:= \sigma(\hat{x}_f), \quad x_2 := \varphi(\hat{x}_f) u_f, \quad k_{wj} > 0 \\
\Lambda_j, \Pi_j &> 0, \quad W_{j,f} = W_{j,*} - W_{j,f} \\
\Pi_j &:= C^T \Lambda_{v_j} C + \Lambda_{W_j}^{-1} \\
N_* &:= \left(CC^T + \delta I_{n\times n}\right)^{-1}, \quad j = 1, 2
\end{aligned} \tag{9}$$

where $P = P^T > 0$ is the solution, if it exists, of the algebraic Ricatti equation given by (10).

$$\begin{aligned}
P_1 \tilde{A}_1^{(0)*} + \left(\tilde{A}_1^{(0)*}\right)^T P_1 + P_1 R_1 P_1 + Q_1 &= 0 \\
\tilde{A}_1^{(0)*} &:= \left(A^{(0)*} - K_1 C\right) \\
R_1 &:= \tilde{W}_1 + \tilde{W}_2 + \Lambda_f^{-1} + \Lambda_{v_1}^{-1} + K_2 \Lambda_{v_2}^{-1} K_2^T \\
Q_1 &:= I_n \Lambda_{v_0} + \Lambda^{-1} + 2\tilde{f}_1 \Lambda_f + \delta^2 \left(\Lambda_{W_1} + \Lambda_{W_2}\right) + Q_0, \quad Q_0 > 0
\end{aligned} \tag{10}$$

here $A^{(0)}$ is the "best" matrix possible value. This was obtained by the training process. In view of (7) structure, when $e_f = \hat{y}_f - C\hat{x}_f = 0$, the ODE (7) should be attended as a differential inclusion in Fillipov sense. This robust adaptive observer is a state estimator more advanced than one which only contains just one linear correction term like Luenberger type observer, since the observer proposed here possesses higher sensibility to external disturbances within a zone with a small estimation error upper bound. As a test of likeness between the mathematical model and a real patient,

different dose strategies have been given such as Gaussian, exponential and pulse dosages, checking in this way that the DNN presents robustness to entrance variations.

2.4. Procedure 3. Control by Hamilton Jacobi and Sliding Modes

With Hamilton Jacobi's Algorithm for the track of trajectory it is possible to simplify the computational spending since this control algorithm uses the estimated states obtained with a DNN. The system's behavior of prospective control is to displace the states such it is possible to make a track of a sign generated by some reference model given by (11).

$$\dot{x}_m = f_m(x_m, t) \tag{11}$$

Let's consider the following semi norm: $|z|_Q^2 = \limsup_{\tau \to \infty} \frac{1}{\tau} \int_0^{\tau} z^T(t) Q z(t) dt$ where $Q = Q^T > 0$. The track of trajectory can be formulated as (12).

$$J_{min} = \min_{u(t)} J, \qquad J = |x - x_m|_{Q_c}^2 + |u|_{\tilde{R}_c}^2 \tag{12}$$

Then, for any $\eta > 0$, we obtain (13):

$$J \le (1+\eta)|x - \hat{x}|_{Q_c}^2 + (1+\eta^{-1})|\hat{x} - x_m|_{Q_c}^2 + |u_t|_{\tilde{R}_c}^2 \tag{13}$$

The superior limit of the term $|x - \hat{x}|_{Q_c}^2$ has been reached by the corresponding states estimator. Selecting $\tilde{R}_c = (1+\eta^{-1})R_c$ we can formulate again the one control objective as follows, minimizing of being possible, the term 13. For this purpose, let us define the error of state trajectories as (14).

$$\Delta_m = \hat{x} - x_m \tag{14}$$

In agreement with 12, in order to carry out the pursuit of the system, the DNN as observer is presented as (15).

$$\dot{x} = F_0(x,t) + F_1(x,t)u \tag{15}$$

For the desire dynamics and well-known 11, the tracking trajectory is completed by the Hamilton Jacobi and Sliding Mode Control (HJSMC) described by (16).

$$u^{HJ} := F_1^{+}(x,t)[-\alpha \cdot SIGN(x - x_m) + (f_m(x_m,t) - F_0(x,t))] \tag{16}$$

where $\alpha > 0$. In this case, when considering the DNN as observer structure, we obtain: $F_0(\hat{x},t) = A\hat{x} + W_1\sigma(\hat{x}) + K_1(y_t - C\hat{x}_t) + K_2 SIGN(y_t - C\hat{x}_t)$ and $F_1(\hat{x},t) = W_2\varphi(\hat{x})$.

3. Numerical Simulation Results

In this section is presented the obtained simulation results, which are associated with the identification, estimation and control processes, showing the importance of computational programming in the study of hepatitis C viral dynamics.

3.1. Estimate Simulation Results

The following results show how the DNN estimator is able to track the trajectories in spite of the fact that the entry changes, which prove by this way the robustness about input variations. The first chart (Fig. 2) considers a dosage strategy called Gaussian which show to have similar way than the one were obtained when exponential dosage strategy were applied to the mathematical model.

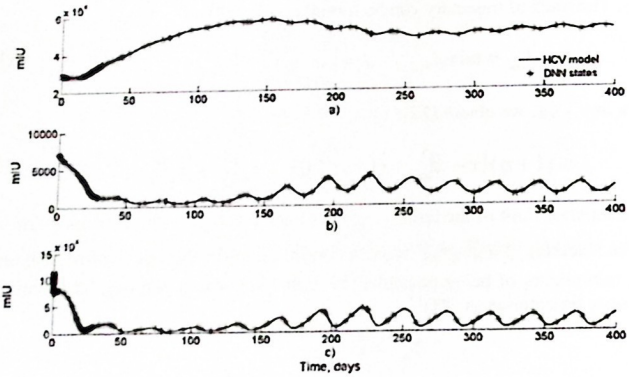


Fig. 2. Results when the treatment is considered as Gaussian signal. a) Target cells concentration, b) Infected cells concentration and c) Viral Load.

3.2. Control Simulation Results

The aim of this work is to control the viral load in a patient infected with HCV by means of the use and application of automatic control algorithms as HJT and SMA. These techniques are allowed to give the exact dose of IFN $\alpha - 2b$ that should be applied to the patient with this illness. In order to carry out this objective, it was necessary to have knowledge in their entirety HCV model. This complete knowledge was obtained by using a neuroobserver based on DNN. Once obtained that, is necessary to have a reference that is adapted to complete the goal, in this paper a reference model

was selected with the intention of eliminating all HCV and all infected cells of the patient's body in a few days as is shown in the figure 3.

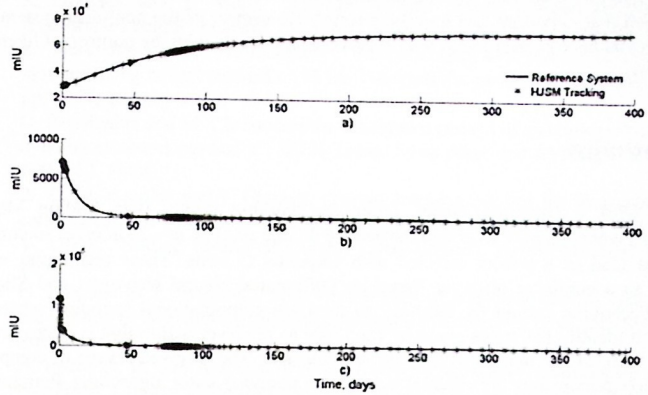


Fig. 3. Simulation results of the control process. a) Target cells concentration, b) Infected cells concentration and c) Viral Load.

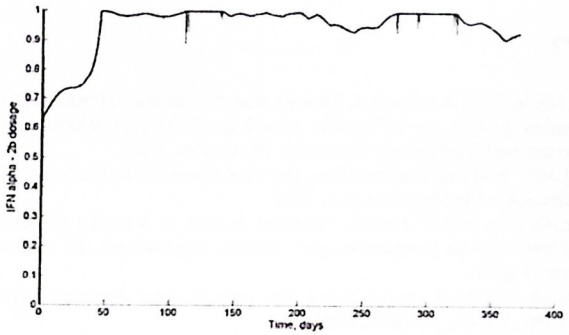


Fig. 4. IFN $\alpha - 2b$ dosage obtained by the HJSMC in order to eliminate the HCV.

The computational algorithms proposed in this paper have demonstrate to be able to control this nonlinear system, a prove of that is presented in figure 3 where it is evident that HJSMC can reconstruct the desire IFN $\alpha - 2b$ patients response eliminating in approximately 2 months more than 80% of HCV and infected productive cells.

To achieve that, the HJSMC provide the HCV model, which is supposed to have similar response than real patient, a strategy dosage the one that is shown in figure 4. This dosage has a maximum value around *25mIU*. This dosage would be the same to each patient provided they had similar response; nevertheless, in this work it has been considered that important fact into the interference vectors of the nonlinear system (2) these vectors have been selected with relatively high levels with the purpose of likening the reality.

4. Conclusion

In this work a Hamilton Jacobi Technique working jointly with Sliding Modes Approach was suggested to obtain the precise dosage on IFN $\alpha - 2b$ in order to control the viral load of a patient infected with Hepatitis C virus. These techniques were applied to a nonlinear observer based on Differential Neural Networks and Sliding Mode Technique getting by this way to diminish computational spending when no having to identify the states given by the DNN as observer action that is necessary to be able to check the nonlinear observer convergence. The received results of computer simulation demonstrate the efficient capacity of applying these algorithms. Perhaps, in a near future, this computational tools will be used in real medical procedures to be able to control, the, until now no controllable, hepatitis C, which has ended up being an important threat for the society.

References

[1] R. F. Miranda, N. Aguilar, J. I. Chairez, and A. Cabrera, "Hepatitis C dynamics' estimation process by differential neural networks", in World Congress on Computational Intelligence, Vancouver BC Canada, 2006.

[2] J. Layden, "Mathematical modeling and viral dynamics of the hepatitis c virus", in mathematical Biology Seminar, 2002.

[3] F. Castiglione and B. Piccoli, "Optimal control in a model of dendritic cell transfection cancer immunotherapy", Istituto Applicazioni del Calcolo (IAC), August 27 2004.

[4] G. Swan, "Optimal control using the verhulst-pearl equation", Bull. Math. Bio.(1986), pp. 48, 381–404, 1990.

[5] T. Burden, J. Ernstberger, and K. Fister, "Optimal control applied to immunotherapy", Discrete and Continuos Dymanical Systems, vol. 4, pp. 135–146, February 2004.

[6] A. Neumann and N. Lam, "Hepatitis c viral dynamics in vivo and the antiviral efficacy of interferon alfa therapy", Science, pp. 103–107, 1998.

[7] M. Diago and R. Planas, "Manual de Hepatitis C Aspectos Biológios, clínicos y terapéuticos", Médica Panamericana, 2004.

[8] V. De Vita, S. Hellman, and S. Rosenberg, "Cancer Principles And Practice of Oncology", Lippincott Williams Wilkins, 2001.

[9] B. Katzung, "Farmacología Básica y Clínica". Manual Moderno, 2005.
[10]. S. Haykin, "Neural Networks. A comprehensive Foundation", New York: IEEE Press, 1994.
[11] S. Douglas and T. Meng, "Linearized least square training of multilayer feedforward neural networks", in Joint Conference on Neural Networks, 1991.
[12] K. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks", IEEE Transactions on Neural Networks, vol. 1, pp. 4–27, January 1990.
[13] G. Rovithakis and M. Christodoulou, "Adaptive control of unknown plants using dynamical neural networks", IEEE Trans. Systs..Man and Cyben., vol. 24, pp. 400–412, 1994.
[14] A. C. N. Aguilar and I. Chairez, "Neuro tracking control for immunotherapy cancer treatmen", in World Congress on Computational Intelligence, 2006.
[15] V. Utkin, "Sliding Modes in Control Optimization". Berlin: Springer Verlag, 1992.
[16] H. Zak and B. L. Walcott, "State observation of nonlinear control systems via the method of lyapunov in zinober Deterministic Control of Uncertain Systems", Stevenage UK, 1990.
[17] A. Poznyak, E. Sanchez, and W. Yu, "Differential Neural Networks for Robust Nonlinear Control (Identification, state Estimation and trajectory Tracking)". Worl Scientific, 2001.
[18] J. Riezu, "PCR en el diagnóstico virológico del virus de la hepatitis c", Departamento de Bioquímica y Biología Molecular. Universidad de Navarra, 2004.

Associative Memories

Associative Memory: Information and Topology

D. Dominguez, K. Koroutchev, E. Serrano and F.B. Rodriguez

EPS, Universidad Autonoma de Madrid,
Cantoblanco, Madrid, 28049, Spain
david.dominguez@uam.es

(Paper received on August 01, 2006, accepted on September 25, 2006)

Abstract. A neural network works as an associative memory device if it has large storage capacity and the quality of the retrieval is good enough. The learning and attractor abilities of the network both can be measured by the mutual information (MI), between patterns and retrieval states. This paper deals with a search for an optimal topology, of a Hebb network, in the sense of the maximal MI. We use small-world topology. The connectivity γ ranges from an extremely diluted to the fully connected network; the randomness ω ranges from purely local to completely random neighbors. It is found that, while stability implies an optimal $MI(\gamma, \omega)$ at $\gamma_{opt}(\omega) \to 0$, for the dynamics, the optimal topology holds at certain $\gamma_{opt} > 0$ whenever $0 \leq \omega < 0.3$.

1 Introduction

The collective properties of attractor neural networks (ANN), such as the ability to perform as an associative memory, has been a subject of intensive research in the last couple of decades[1], dealing mainly with fully-connected topologies. More recently, the interest on ANN has been renewed by the study of more realistic architectures, such as small-world [3] or scale-free [13] models. The storage capacity α_c and the overlap m with the memorized patterns are the most used measures of the retrieval ability for the Hopfield-Hebb networks[4]Comparatively less attention has been paid to the study of the mutual information (MI) between stored patterns and the neural states[5][6], although neural networks are information processing machines.

A reason for this relatively low interest is twofold: on the one hand, it is easier to deal with the global parameter $m[\sigma, \xi]$, than with $MI[p(\sigma|\xi)]$, a function of the conditional probability of neuron states σ given the patterns ξ. This can be solved for the so called *mean-field networks* which satisfy the law of large numbers, hence MI is a function only of the macroscopic parameters m, and the load rate $\alpha = P/K$ (where P is the number of uncorrelated patterns, and K is the neuron connectivity). On the other hand, the load α is enough to measure the information if the overlap is close to $m \sim 1$, since in this case the information carried by any single binary neuron is almost 1 bit. It is true for a fully-connected (FC) network, for which the critical $\alpha_c^{FC} \sim 0.138$ [4], with $m_c^{FC} \sim 0.97$ (with a sharp transition to $m \to 0$ for larger $\alpha \geq \alpha_c$): in this case, the information rate is about $i_c^{FC} \sim 0.131$, as can be seen in the left panel of Fig.1. There we show the

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 39-48

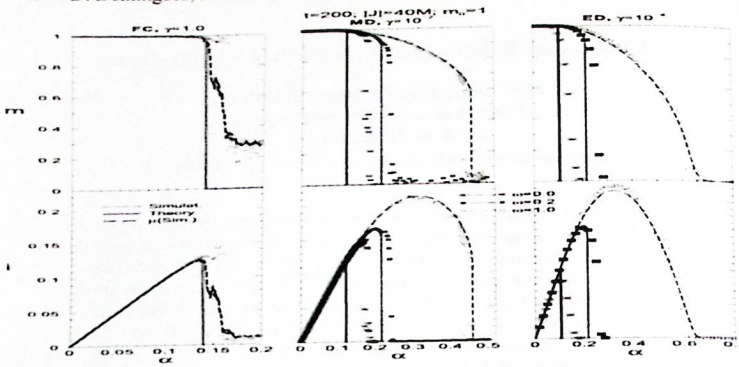


Fig. 1. *The overlap m and the information i vs α for different architectures: fully-connected, $\gamma^{FC} = 1.0$ (left), moderately-diluted, $\gamma^{MD} = 10^{-2}$ (center) and extremely-diluted, $\gamma^{ED} = 10^{-4}$ (right). Symbols represents simulation with initial overlap $m^0 = 1$ and $|J| = 40M$, with local (stars, $\omega = 0.0$), small-world (filled squares, $\omega = 0.2$), and random (circles, $\omega = 1.0$) connections. Lines are for theoretical results: solid, $\omega = 0.0$, dotted, $\omega = 0.2$, and dashed, $\omega = 1.0$. In left, dashed line means averaging the simulation.*

overlap (upper) and information for several architectures. However, in the case of diluted networks the transition is smooth. In particular, the random extremely diluted (RED) network has load capacity $\alpha_c^{RED} \sim 0.64$[8] but the overlap falls continuously to $m_c^{RED} \sim 0$, which yields null information at the transition, $i_c^{RED} \sim 0.0$, as seen in right panel of Fig.1 (dashed line). Such indetermination shows that one must search for the value of α_{max} corresponding to the maximal information $MI_{max} \equiv MI(\alpha_{max})$, instead of α_c.

We address the problem of searching for the optimal topology, in the sense of maximizing the mutual information. Using the graph framework [2], one can capture the main properties of a wide range of neural systems, with only 2 parameters: $\gamma \equiv K/N$, which is the average rate of links per neurons, where N is the network size, and ω, which controls the rate of random links (among all neighbors). When γ is large, the clustering coefficient is large ($c \sim 1$) and the mean-length-path between neurons is small ($l \sim \ln N$), whatever ω is. When γ is small, then if ω is too small, $c \sim 1$ and $l \sim N/K$, but if it is about $\omega \sim 0.1$, the network behaves again as if $\gamma \sim 1$, with $c \sim 1$ and $l \sim \ln(N)$. This region, called small-world (SW), is rather usefull when one is interested to built networks where the information transmition is fast and efficient, with high capacity in presence of significant noise, but do not wants to spent too much wiring.Small-world networks may model many biological systems [12]. For

instance, in a brain local connections dominate in intracortex, while there are a few intercortical connections [11].

In Fig.1 we show the overlap (upper) and information for several architectures. In the left panel, it is seen that the maximum information rate, $i \equiv MI/(K.N)$, of FC network is about $i_{max}^{FC} = 0.135$, while in the right panel, we show extremely-diluted networks (ED). The RED network ($\omega = 1.0$) has $i_{max}^{RED} \sim 0.223$. The right panel of Fig.1 plot also the overlap and the information for the local extremely diluted network (LED, $\omega = 0.0$), with $i_{max}^{LEC} = 0.0855$, and a small-world extremely diluted network (SED, $\omega = 0.2$), with $i_{max}^{SED} = 0.165$. We see that the ED transitions are smooth. The central panel of Fig.1 plot moderately diluted (MD) networks, which are commented later. Theoretical results fit well with the simulations, except for small ω, where theory underestimate it. Previous works about small-world attractor neural networks [10] studied only the overlap $m(\alpha)$, so no result about information were known.

Our main goal in this work is to solve the following question: how does the maximal information, $i_{max}(\gamma, \omega) \equiv i(\alpha_{max}; \gamma, \omega)$ behaves with respect to the network topology? To our knowledge, up to now, there were no answer to this question. We will show that, near to the stationary retrieval states, for every value of the randomness $\omega > 0$, the extremely-diluted network, performs the best, $\gamma_{opt} \to 0$. However, regarding the attractor basins, starting far from the patterns, the optimal topology holds for moderate γ_{opt}. For instance, if transients are taken in account, values of $\omega \sim 0.1$ lead to an optimal $i_{opt}(\gamma) \equiv i_{max}(\gamma_{opt}, \omega)$ with $\gamma_{opt} \sim 10^{-2}$.

The structure of the paper is the following: in the next section we review the information measures used in the calculations; in Sec.3, we define the topology and neuro-dynamics model. The results are shown in Sec.4, where we study retrieval by theory and simulation (with random patterns and with images); conclusions are drawn in last section.

2 The Information Measures

2.1 The Neural Channel

The network state at a given time t is defined by a set of binary neurons, $\boldsymbol{\sigma}^t = \{\sigma_i^t \in \{\pm 1\}, i = 1, ..., N\}$. Accordingly, each pattern $\boldsymbol{\xi}^\mu = \{\xi_i^\mu \in \{\pm 1\}, i = 1, ..., N\}$, is a set of site-independent random variables, binary and uniformly distributed: $p(\xi_i^\mu = \pm 1) = 1/2$. The network learns a set of independent patterns $\{\boldsymbol{\xi}^\mu,\ \mu = 1, ..., P\}$.

The task of the neural channel is to retrieve a pattern (say, $\boldsymbol{\xi}$) starting from a neuron state which is inside its attractor basin, $B(\boldsymbol{\xi})$, *i.e.*: $\boldsymbol{\sigma}^0 \in B(\boldsymbol{\xi}) \to \boldsymbol{\sigma}^\infty \approx \boldsymbol{\xi}$. This is achieved through a network dynamics, which couples neighbor neurons $\sigma_i . \sigma_j$ by the *synaptic matrix* $\mathbf{J} \equiv \{J_{i,j}\}$ with cardinality $|\mathbf{J}| = N \times K$.

2.2 The Overlap

For the usual binary non-biased neurons model, the relevant order parameter is the *overlap* between the neural states and a given pattern:

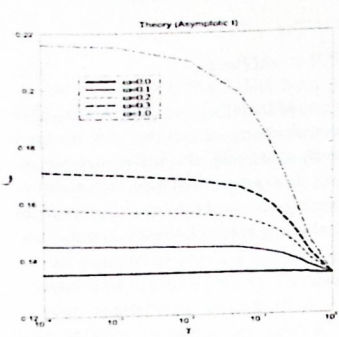


Fig. 2. Maximal information $i_{max} = i(\alpha_{max})$ vs γ. Theoretical results for the stationary states, with several values of randomness ω.

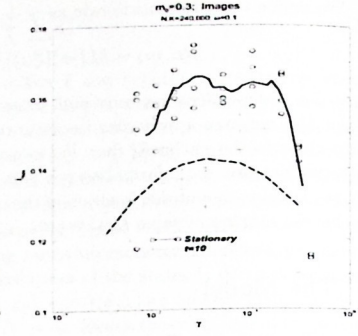


Fig. 3. i_{max} vs γ. Simulation with $\omega = 0.1$ and $m^0 = 0.3$. Dynamics stop at $t = 10$ (Plus dots, Solid line) or at $t = 100$ (Circles, Dashed line).

where x is a normalized random variable and T is the temperature-like environmental noise. In the case of symmetric synaptic couplings, $J_{ij} = J_{ji}$, an energy function $H_s = -\sum_{(i,j)} J_{ij}\sigma_i\sigma_j$ can be defined, whose minima are the stable states of the dynamics Eq.(5).

In the present paper, we work out the asymmetric network by simulation (no constraints $J_{ij} = J_{ji}$). The theory was carried out for symmetric networks. As it is seen in Fig.1, theory and simulation shows similar results, except for local networks (theory underestimate α_{max}, where the symmetry may play some role. We restrict our analysis also for the deterministic dynamics ($T = 0$). The stochastic macro-dynamics comes from the extensive number of learned patterns, $P = \alpha K$.

4 Results

We studied the information for the stationary and dynamical states of the network were studied as a function of the topological parameters, ω and γ. A sample of the results for simulation and theory is shown in Fig.1, where the stationary states of the overlap and information are plotted for the FC, MD and ED architectures. It can be seen that information increases with dilution and with randomness of the network. A reason for this behavior is that dilution decreases the correlation due to the interference between patterns. However, dilution also increases the mean-path-length of the network, thus, if the connections are local, the information flows slowly over the network. Hence, the neuron states can

be eventually trapped in noisy patterns. So, i_{max} is small for $\omega \sim 0$ even if $\gamma = 10^{-4}$.

4.1 Theory: Stationary States

Following to the Gardner calculations[8], at temperature T=0 the MFN approximation gives the fixed point equations:

$$m = \mathrm{erf}(m/\sqrt{r\alpha}), \tag{6}$$

$$\chi = 2\varphi(m/\sqrt{r\alpha})/\sqrt{r\alpha}; \tag{7}$$

$$r = \sum_{k=0}^{\infty} a_k(k+1)\chi^k, \;\; a_k = \gamma Tr[(\mathbf{C}/K)^{k+2}] \tag{8}$$

with $\mathrm{erf}(x) \equiv 2\int_0^x \varphi(z)dz$, $\varphi(z) \equiv e^{-z^2/2}/\sqrt{2\pi}$. The parameter a_k is the probability of existence of cycle of length $k+2$ in the connectivity graph. The a_k can be calculated either by using Monte Carlo [14], or by an analytical approach, which gives $a_k \sim \sum_m \int d\theta[p(\theta)]^k e^{im\theta}$, where $p(\theta)$ is the Fourier transform of the probability of links, $p(C_{ij})$. For an RED and FC networks one recover the known results for $r^{RED} = 1$ and $r^{FC} = 1/(1-\chi)^2$ respectively [1].

The theoretical dependence of the information on the load, for FC, MD and ED networks, with local, small-world and random connections, are plotted in the fat lines in Fig.1. A comparison between theory and simulation is also given in Fig.1. It can be seen that both results agree for most $\omega > 0$, but theory fails for $\omega = 0$. One reason is that theory uses symmetric constraint, while simulation was carried out with asymmetric synapsis. Figure 2 shows their maxima $i(\alpha_{max})$ vs. the parameters (ω, γ). It is seen that the optimal is at $\omega \to 1, \gamma \to 0$. This implies that the best topology for information (stationary states) is the extreme diluted network, with purely random connectivity.

4.2 Simulation: Attractors and Transients

We have studied the behavior of the network varying the range of connectivity γ and randomness ω. We used Eq.(5). Both local and random connections are asymmetric. The simulation was carried out with $N \times K = 36 \cdot 10^6$ synapses, storing an adjacency list as data structure, instead of J_{ij}. For instance, with $\gamma \equiv K/N = 0.01$, we used $K = 600, N = 6 \cdot 10^4$. In [10] the authors use $K = 50, N = 5 \cdot 10^3$, which is far from asymptotic limit.

We studied the network by searching for the stability properties and transients of the neuron dynamics. To look for stability, we started the network at some pattern (with initial overlap $m^0 = 1.0$), and wait until it stays or leave it after a flag time step $t = t_f$ (unless it converges to a fixed point m^* before $t = t_f$). When we check transients, we start with $m^0 = 0.1$, and stop the dynamics at the time t_f. Usually, $t_f = 20$ parallel (all neurons) updates is a large enough delay for retrieval. Indeed in most case far before the saturation, after

$t_f = 4$ the network end up in a pattern, however, near α_{max}, even after $t_f = 100$ the network has not yet relaxed.

In first place, we checked for the stability properties of the network: the neuron states start precisely at a given pattern $\boldsymbol{\xi}^\mu$ (which changes at each learned step μ). The initial overlap is $m_0^\mu = 1.0$, so, after $t_m \leq 20$ time steps in retrieving, the information $i(\alpha, m; \gamma, \omega)$ for final overlap is calculated. We plot it as a function of α, and its maximum $i_{max} \equiv i(\alpha_{max}; \gamma, \omega)$ is evaluated. Second, we checked for the retrieval properties: the neuron states start far from a learned pattern, but inside its basin of attraction, $\boldsymbol{\sigma}^0 \in B(\boldsymbol{\xi}^\mu)$. The initial configuration is chosen with distribution: $p(\sigma_i^0 = \pm \xi_i^\mu | \xi_i^\mu) = (1 \pm m^0)/2$, for all neurons (so we avoid a bias between local/random neighbors). The initial overlap is now $m^0 = 0.1$, and after $t_f \leq 20$ steps, the information $i(\alpha, m; \gamma, \omega)$ is calculated.

Each maximal $i_{max}(\gamma; \omega)$ is plotted in Fig.4. We see that, for intermediate values of the randomness parameter $0 \leq \omega < 0.3$ there is an optimal information respect to the dilution γ, if dynamics is truncated. We observe that the optimal $i_{opt} \equiv i_{max}(\gamma_{opt}; \omega)$ is shifted to the left (stronger dilution) when the randomness ω of the network increases. For instance, with $\omega = 0.1$, the optimal is at $\gamma \sim 0.020$ while with $\omega = 0.2$, it is $\gamma \sim 0.005$. This result does not change qualitatively with the flag time, but if the dynamics is truncated early, the optimal γ_{opt}, for a fixed ω, is shifted to more connected networks. However, the behavior depends strongly on the initial condition: respect to $m_0 = 0.1$, where the maximal are pronounced, with $m_0 = 1.0$, the dependence on the topology becomes almost flat. We see also that for $\omega \geq 0.3$ there is no intermediate optimal topology. It is worth to note that the simulation converges to the theoretical results if $m_0 = 1.0$ when $t \to \infty$.

One can understand this non-monotonic behavior of the information in terms of the basins of attraction. Random topologies have very deep attractors, specially if the network is diluted enough, while regular topologies almost lose their retrieval abilities with dilution. However, since the basins becomes rougher with dilution, then network takes longer to reach the attractor. Hence, the competition between depth-roughness is won by the more robust MD networks.

4.3 Simulation with Images

The simulations presented so far use artificial patterns randomly generated. In order to check if our results are robust against possibly correlations existent in realistic patterns, we test the algorithm with images. We see that the same non-monotonic behavior for $i_{max}(\gamma)$ is observed here.

We have checked the results by using data derived from the Waterloo image database. We are working with square shaped patches. In order to use Hebb-like non-sparse code binary network and still preserve the structure of the image we process the images preserving the edges, by applying edge filter. Each pixel of the patch represents a different neuron. The number of connections is up to $N \times K = 3 \cdot 10^5$ and the feasible connectivities (more than 3 patterns) are $\gamma > 0.002$.

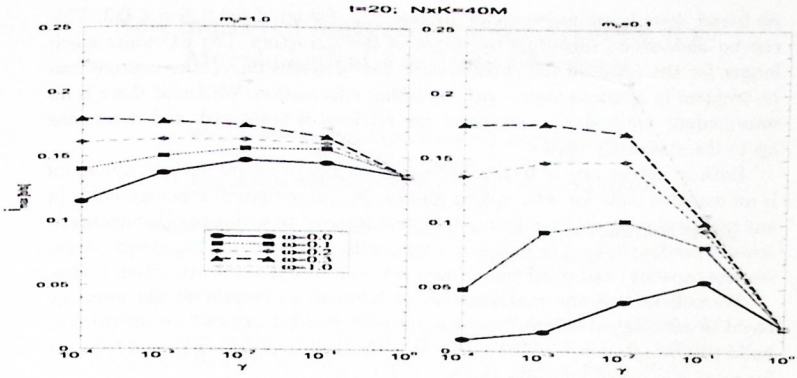


Fig. 4. Maximal information $i_{max} = i(\alpha_{max})$ vs γ, for simulations with $N.K = 4.10^7$, and several ω. Initial overlap $m^0 = 1.0$ (left) and $m^0 = 0.1$ (right); the retrieval stops after $t_f = 20$ steps.

Note that the procedure, strictly speaking, does not guarantee the conditions for the distribution of ξ, because neither $p(\xi = \pm 1)$ is uniform (due to the threshold in large blocks), nor ξ_i are uncorrelated (due to image edges).

We are choosing at random the origin of the patch and the image to be used from the available 12 images. The topology of the network is a ring with small world topology. The results of the simulation, using Chen filter, are shown in Fig.3. The optimal connectivity with $\omega = 0.1$ and $t_f = 10$ is found to be $\gamma_{opt} \sim 0.03$. The fluctuation now are much larger than with random patterns, due to correlation and small network size. In the stationary states, $t_f \to \infty$, the optimal connectivity remains at $\gamma_{opt} \sim 0.03$, with $i_{opt} \sim 0.165$. The results agree qualitatively with simulation for random patterns, Fig.4, where the initial overlaps are $m^0 = 0.1$ and $m^0 = 1.0$ (in Fig.3 it is always $m^0 = 0.3$).

5 Conclusions

In this paper we have studied the dependence of the information capacity with the topology for an attractor neural network. We calculated the mutual information for a Hebb model, for storing binary patterns, varying the connectivity (γ) and randomness (ω) parameters, and obtained the maximal respect to α, $i_{max}(\gamma, \omega) \equiv i(\alpha_{max}; \gamma, \omega)$. Then we look at the optimal topology, γ_{opt} in the sense of the information, $i_{opt} \equiv i_{max}(\gamma_{opt}, \omega)$. We presented stationary and transient states. The main result is that larger ω always leads to higher information i_{max}.

From the stability calculations, the stationary optimal topology, is the extremely diluted (RED) network. Dynamics shows, however, that this is not true:

we found there is an intermediate optimal γ_{opt}, for any fixed $0 \leq \omega < 0.3$. This can be understood regarding the shape of the attractors. The ED waits much longer for the retrieval than more connected networks do, so the neurons can be trapped in spurious states with vanishing information. We found there is an intermediate optimal γ_{opt}, whenever the retrieval is truncated, and it remains up to the stationary states.

Both in nature and in technological approaches to neural devices, dynamics is an essential issue for information process. So, an optimized topology holds in any practical purpose, even if no attemption is payed to wiring or other energetic costs of random links. The reason is a competition between the broadness (larger storage capacity) and roughness (slower retrieval speed) of the attraction basins.

We believe that the maximization of information respect to the topology could be a biological criteria (where non-equilibrium phenomena are relevant) to build real neural networks. We expect that the same dependence should happens for more structured networks and learning rules.

Acknowledgments Work supported by grants TIC01-572, TIN2004-07676-C01-01, BFI2003-07276, TIN2004-04363-C03-03 from MCyT, Spain.

References

1. Hertz, J., Krogh, J., Palmer, R.: Introduction to the Theory of Neural Computation. Addison-Wesley, Boston (1991)
2. Strogatz, D., Watts, S.: Nature **393** (1998) 440
3. Masuda, N. and Aihara, K. Biol. Cybernetics, **90**: 302 (2004)
4. Amit, D., Gutfreund, H., Sompolinsky, H.: Phys. Rev. A **35** (1987) 2293
5. Perez-Vicente, C., Amit, D.: J. Phys. A, **22** (1989) 559
6. Dominguez, D., Bolle, D.: Phys. Rev. Lett **80** (1998) 2961
7. Bolle, D., Dominguez, D., Amari, S.: Neural Networks **13** (2000) 455
8. Canning, A. and Gardner, E. *Partially Connected Models of Neural Networks*, J. Phys. A, **21**, 3275-3284, 1988
9. Kupermann, M. and Abramson, G. Phys. Rev. Lett. **86**: 2909, 2001
10. McGraw, P.N. and Menzinger, M. Phys. Rev. E **68**: 047102-1, 2003
11. Rolls, E., Treves, A., Neural Network and Brain Function. Oxford U. Press, 2004
12. Sporns, O. et al., Cognitive Sciences, **8(9)**: 418-425, 2004
13. Torres, J. et al., Neurocomputing, **58-60**: 229-234, 2004
14. Dominguez, D. et al., LNCS **3173**: 14-29, 2004
15. Li, C., Chen, G.: Phys. Rev. E **68** (2003) 52901

New Improved Algorithm for the Training of a Morphological Associative Memory

Ricardo Barrón, Humberto Sossa and Benjamín Cruz
Centro de Investigación en Computación-IPN
Av. Juan de Dios Bátiz esquina con Miguel Othón de Mendizábal
Mexico City, 07738. MEXICO
E-mails: rbarron@cic.ipn.mx, hsossa@cic.ipn.mx, benjamincruz@sagitario.cic.ipn.mx,

(Paper received on September 01, 2006, accepted on September 27, 2006)

Abstract. We describe in this work a new algorithm for the training of an associative memory based on the so-called multi-layered morphological perceptron with maximal support neighborhoods. We describe how to improve the original algorithm by adding new steps at the training phase. We have performed several experiments with real patterns where we show the superiority of the new proposal. We also give formal conditions for correct classification.

1. Introduction

Neural networks are a computational alternative to solve problems where it is difficult to find or there is no an algorithmic solution. Based on the functioning of the human nervous system, lots of researchers have proposed different neural processing models. The study of the internal structure of the neural cells has revealed that all cells have the same simple structure, independently of their size and shape (Fig. 1).

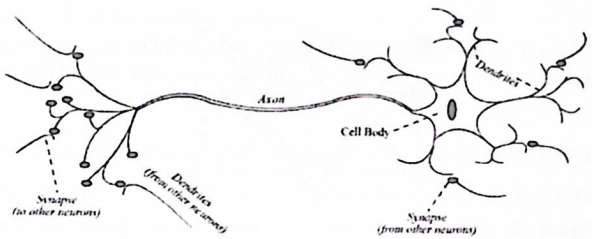


Figure 1. Structure of a biological neuron.

Information from this cell voyages through the signals that neurons send to other neurons through their dendrites. It is believed that the cellular body adds up the received signals; when enough inputs are available a discharge is produced. Initially, this discharge occurs in cellular body, it propagates between the axon until the synapses that sends a new signal to the other neurons.

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 49-59

One of the drawbacks of this proposal to build the supports is that if two of these supports are too close, the radius of their neighborhoods reduces drastically.

In [7] it is proposed another method to increase the neighborhoods of the supports by means of the kernels' method. According to [7], with this the range of permissible noise is increased. However, this method makes expensive the computational cost and besides it imposes restrictions over the patterns very difficult to get.

In the following section we describe a new way to train this kind of associative memories that will allow, as we will see, to obtain wider support neighborhoods.

3. Proposed training algorithm

If we suppose that a pattern is represented in terms of n describing features, then at each coordinated axis we can compute the variation obtained per feature by ordering all components. By computing the average of variability per pattern it is possible to define a threshold. For practical purposes, this threshold allows to consider if two patterns can be considered to be the same from the point of view of one of their components. This way we avoid having very tiny supports with respect to the coordinated axis. This way the drawback of the algorithm described in last section is surpassed.

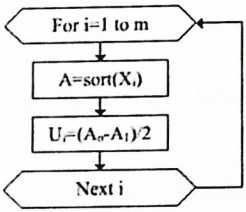


Figure 3. Flowchart to get average variation threshold.

The algorithm to find the average variation threshold by axis is observed in Fig. 3. In this case, X is a matrix of $n \times m$. At j-th column it is j-th pattern, while at i-th line are the n features of the i-th axis of each pattern. In U_i it is the average variation threshold for the i-th axis. The key point for the training of the multi- layered morphological perceptron consists on constructing the supports for each pattern. In general, these supports must be disjoint two by two to avoid that two patterns are assigned to the same output.

In some cases we have patterns that are too close: the neighborhoods of both patterns present occlusions. In this particular case we follow the original algorithm only for the component where occlusion occurs. Evidently if all patterns are too close the proposal is the same as the original algorithm.

With these arrangements we have disjoints supports two by two. This is fulfilled in the proposed algorithm thanks to the following:

Proposition 2. *Let* Ω_i *and* Ω_j *two arbitrary supports corresponding to different patterns in* $\mathbf{R}^n$, *built according to average variation threshold, then it holds that* $\Omega_i \cap \Omega_j = \varnothing$.

Proof. Let x^i and x^j the corresponding patterns to neighbors supports Ω_i and Ω_j, respectively, then we have two cases:

a. If it holds that $x_k^i - x_k^j < U_k, \forall_{k=1,\ldots,n}$, with U_k is the average variation. If this holds, we apply the original algorithm and by proposition 1 it holds that $\Omega_i = \Omega_j$.

b. If $\exists_{k=1,\ldots,n}$ such that $x_k^i - x_k^j > U_k$ would imply that at $k-th$ coordinate, the support does not coincide and on the axis they are disjoint, thus they are disjoint in $\mathbf{R}^n$. ■

One advantage of this enhancement is that neighborhoods allowing more noise to be added to the patterns will be enlarged.

4. Numerical Examples

Example No. 1. Let the following set of key patterns in $\mathbf{R}^n$:

$$\left\{x^1 = \begin{bmatrix} 1 \\ -4 \end{bmatrix}, x^2 = \begin{bmatrix} 1 \\ -5 \end{bmatrix}, x^3 = \begin{bmatrix} -3 \\ 5 \end{bmatrix}, x^4 = \begin{bmatrix} -4 \\ -4 \end{bmatrix}, x^5 = \begin{bmatrix} 4 \\ 2 \end{bmatrix}, x^6 = \begin{bmatrix} 5 \\ 3 \end{bmatrix}\right\} \tag{4}$$

a) Solution obtained by means of algorithm proposed in [7]:

By applying the algorithm proposed in [7], according to equation (2) $\alpha = 0.5$. Fig. 4 shows the neighborhoods obtained when using this value of α.

b) Solution obtained by means of algorithm proposed in this paper (first variant):

When using the algorithm to get the maximal support neighborhoods, by applying the flowchart shown in Fig. 3, we get the threshold value U as:

$$U = \begin{bmatrix} 1.5 \\ 1.6 \end{bmatrix}. \tag{5}$$

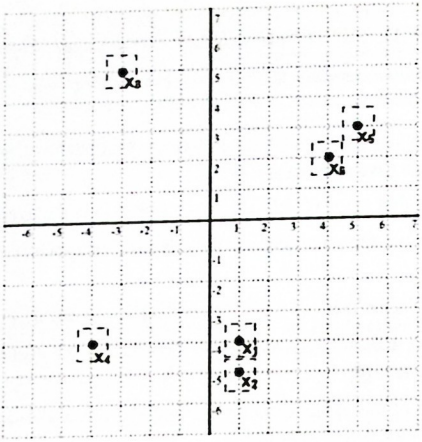


Figure 4. Neighborhoods obtained when algorithm proposed in [7] is used.

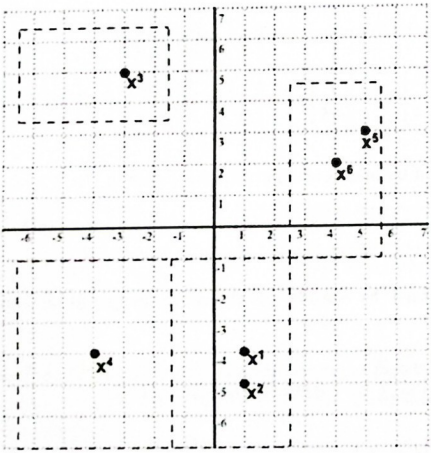


Figure 5. Neighborhoods obtained when using first variant of the proposed algorithm.

The neighborhoods obtained by using this threshold value are shown in Fig. 5. In this case we do not take into account the possible occlusions that can occur. We use this approach in order to compare with the results with the new approach. The differences between the proposal and original one can be immediately appreciated. In this second case the range on noise for each pattern, as can be appreciated, is bigger.

Although, in some cases when we want recognize original patterns (patterns that are not affected by noise) we get mistakes during classification because two or more patterns are assigned at same support neighborhood.

c) Solution obtained by means of algorithm proposed in this paper (second variant):

We apply the same procedure used by the first variant, but in this case the neighborhoods present occlusions, we apply the original algorithm. The threshold is the same given by equation (5). We use the value of $\alpha = 0.5$.

The neighborhoods obtained are shown in Fig. 6. We can observe that the neighborhoods do not present occlusions and the supports for noise are bigger.

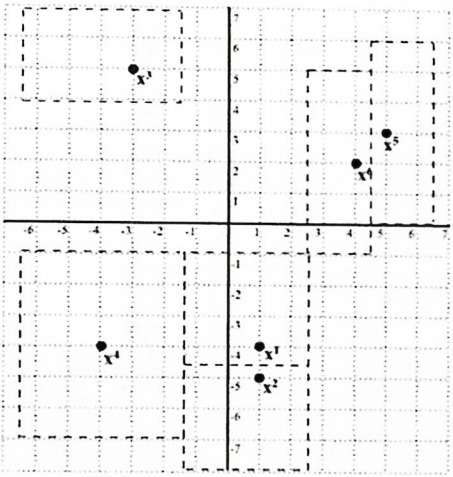


Figure 6. Neighborhoods obtained when using second variant of the proposed algorithm.

5. Experiments

To verify the efficiency of the proposed two variants in this work with respect to the classical algorithm described in [7] used to build neighborhood supports, we have done

the following simple test. It consisted on recognizing the images shown in Fig. 7. These images are binary of 37×31 elements. These patterns were perturbed with noise from 5% to 15% in steps of 5%.

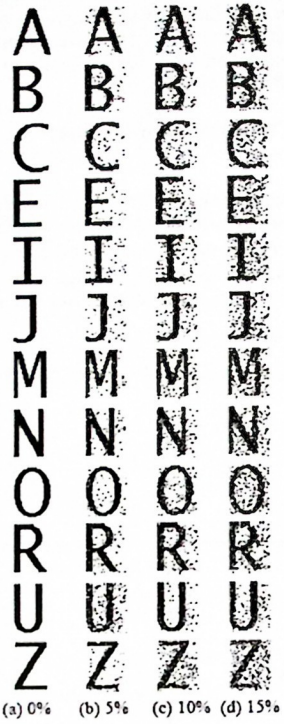


Figure 7. Images used to test the proposal. (a) Original images. (b), (c),(d) Altered images with 5%, 10% and 15% of noise, respectively.

The input key patterns where formed by describing each image by means of known Hu invariants ([10], [11]). Fig. 8 shows graphically the results obtained when using the algorithm proposed in [7], and the algorithm proposed in this paper to the images shown in Fig. 7(a). In this case, as can be appreciated in all cases, the original proposal presents the best performance. The first proposed variant of this paper do not provide does not provide good results.

Figures 9, 10 and 11 show the recognition results obtained when applying the original algorithm and the two variants described in this paper to the distorted versions

of the patterns. From these figures, we can immediately appreciate that in practically any of the cases the original algorithm fails to correctly classify the desired patterns. The new show the best performance.

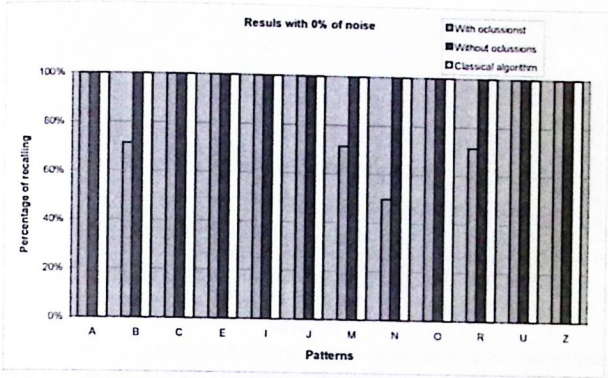


Figure 8. Results when recalling the images of Figure 7(a).

From all of these graphics we can observe that when adding noise to the patterns, even small quantities of it, the original algorithm proposed in [7] fails to recall practically all patterns, while the proposed two variants, although with modest percentage, they arrive to recall several of the desired patterns.

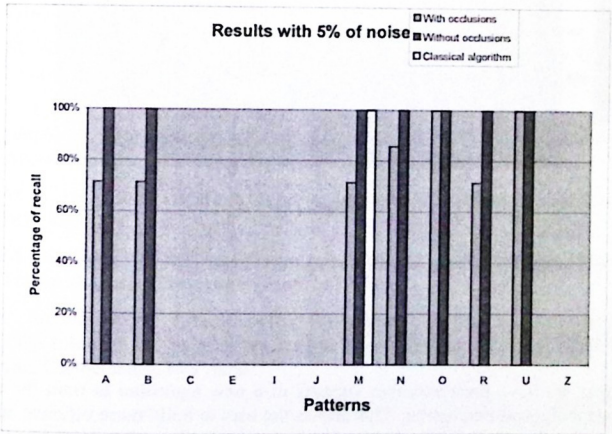


Figure 9. Results when recalling the images of Figure 7(b).

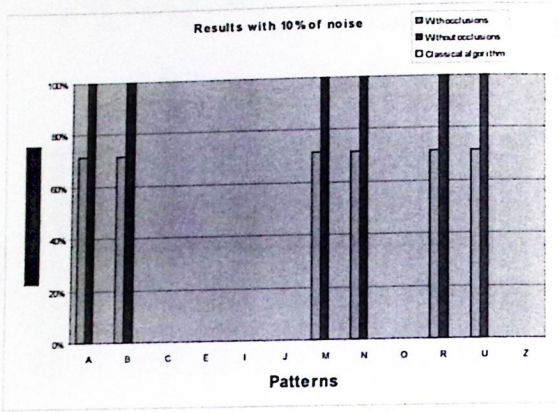


Figure 10. Results when recalling the images of Figure 7(c).

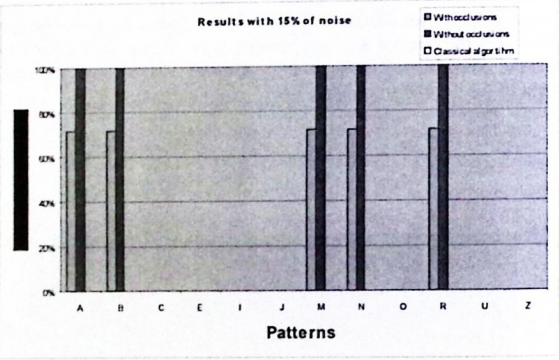


Figure 11. Results when recalling the images of Figure 7(d).

6. Conclusion

In this paper we have presented two variants of a new algorithm to train the multi-layered morphological perceptron. This allows the user to build more efficient support neighborhoods, since the point of view of pattern recall, from the set of key pattern

patterns of the training set of an associative memory in both its auto-associative or hetero-associative way of operation. Experiments have been carried out that show the superiority of the proposed variants proposal with respect to the classical algorithm. Formal conditions for correct classification have also given.

Acknowledgements. This work was economically supported by CGPI-IPN under grants 20050156, 20060517 and CONACYT by means of grant 46805.

References

[1] C. Bishop (1995). Neural Networks for Pattern Recognition. Oxford University Press, Oxford, England.

[2] F. Rossenblatt (1958). The perceptron: a probabilistic model for information storage and organization in the brain, Psychological Review 65:386-408.

[3] F. Rossenblatt (1962). Principles of Neurodinamics: Perceptrons and the theory of brain mechanisms, Spartian Books, Washington, D.C.

[4] G. X. Ritter et al (1996). An introduction to morphological neural networks. In Proceedings of the 13th International Conference on Pattern Recognition, pp. 709-717. Vienna, Austria.

[5] G. X. Ritter et al. (1998). Morphological associative memories. IEEE Transactions on Neural Networks, 9:281-293.

[6] G. X. Ritter et al (1997). Morphological perceptrons. ISAS'97, Intelligent Systems and Semiotics, Gaithersburg, Maryland.

[7] G. X. Ritter (2004). A new auto-associative memory based on lattice algebra, Proc. 9th CIARP'04, La Havana, Cuba 148-155.

[8] M. Kishan, K. M. Chilukuri, and R. Sanjay (1997). Elements Of Artificial Neural Network, The MIT Press, Cambridge, Massachusetts, London, England.

[9] L. F. C. Pessoa et al. (1996) Morphological/Rank neural networks and their adaptative optimal image processing, Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, 6:3399-3402.

[10] M. K. Hu (1961). Pattern recognition by moment invariants, Proceedings of the IRE, 49:1428.

[11] M. K. Hu (1962). Visual pattern recognition by moment invariants, IRE Transactions on Information Theory, 179-187.

[12] P. Sussner (1998). Morphological Perceptron Learning. In Proceedings of the 1998 International Symposium on Intelligent Systems and Semiotics, pp. 477-482, Gaithersburg, Maryland.

Efficient Codebooks for Visual Concept Recognition by ASSOMs Activation

Grégoire Lefebvre and Christophe Garcia
France Telecom R&D - Orange - TECH/IRIS/CIM
4, Rue du Clos Courtel
35512 Cesson S´evign´e Cedex – France
{gregoire.lefebvre,christophe.garcia}@orange-ft.com

(Paper received on August 10, 2006, accepted on September 25, 2006)

Abstract. In this paper, we propose a novel method for robustly characterizing and classifying visual concepts, and more precisely for detecting several categories of complex objects in images. Toward this aim, we propose a scheme that relies on Adaptive-Subspace Self-Organizing Maps (ASSOMs). Robust local signatures are first extracted from training object images and projected into specialized ASSOM networks. The extracted local signatures activate several neural maps producing activation energies. These activation energies are then fused into global feature vectors representing the object images. Object recognition is then performed via a supervised SVM (Support Vector Machine) classification. A multiscale search approach completes the system in order to obtain the object localization and identification in complex scenes. The proposed method allows a good detection rate of 85.08% for the PASCAL 2005 challenge1, composed of 689 complex real world images, containing four different objects undergoing strong variations in shapes, sizes, poses and illumination conditions.

1. Introduction

According to several psycho-visual experiments [1], the human vision system performs saccadic eye movements between salient locations to capture image content. Many systems in computer vision are inspired by this observation, in order to describe visual information for image classification or retrieval. In opposition to global approaches, for which a signature is computed by considering all pixels in the image, local approaches represent image content via a set of local signatures centered on interest points (IP) [2–4], which are extracted on perceptually important areas.

Tversky studies [5] showed that when we compare two images, we detect common and distinct concepts between the regions around the IPs. Our method aims at reproducing these concepts with a codebook learning strategy based on ASSOM activation maps for each category. Visual similarity is then estimated by the distance between different activation histograms.

Our method has been experimented in the context of an object detection task where the good detection rate reaches 85.08% for 689 complex real world images, from the Pascal 2005 Challenge[1], containing four different object categories.

[1] http://www.pascal-network.org/challenges/VOC/voc2005

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 61-71

This paper is organized as follows. In Section 2, we first present our object detection scheme based on ASSOM activation energies. Then, Section 3 demonstrates our system performances with some experimental results. Finally, conclusions are drawn.

2. Object Detection Based on ASSOM Energies

2.1. Proposed Scheme Overview

As outlined by R.O. Duda [6], a classification scheme is generally composed of three main steps: pre-processing, feature extraction and feature classification. In the proposed study, we mainly focus on the two first steps, the last step being performed by a SVM classifier.

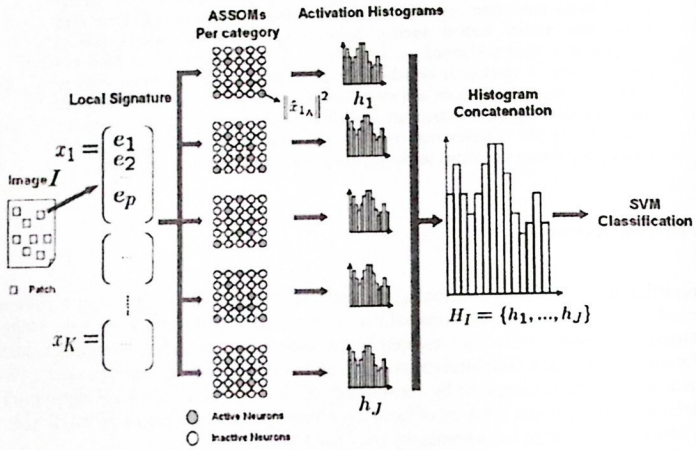


Fig. 1. The Proposed System Architecture.

Our system architecture consists of six steps in the learning phase (see Figure 1):

- We first locate the salient zones with an IP detector [2] mainly on sharp region boundaries.
- Local visual features are then extracted in order to describe the orientation and the regularity of the singularities contained in the different patches around each detected IP.

- Visual feature vectors are fed into specialized ASSOM networks to characterize the main visual prototypes via neural activation maps. These maps synthesize the activation energies for each category.
- Activation energies are represented by activation histograms for each class.
- These histograms are concatenated to build the image global feature vector.
- Finally, a SVM classifier is trained in a supervised way from these discriminative global feature vectors.

2.2 Regularity Foveal Descriptor

Most local descriptors represent the neighborhood around salient points by characterizing edges in this area [7]. To describe edges, gradient orientations and magnitudes are generally used. In a recent study [8], it has been shown that an edge or more generally a singularity can also be efficiently characterized by considering its Hölder exponents.

Definition 1. *$f : [a,b] \rightarrow \mathbb{R}$ is Hölder $\alpha \geq 0$ at $x_0 \in \mathbb{R}$ if $\exists K > 0$, $\delta > 0$ and a polynom P of degree $m = \lfloor \alpha \rfloor$: $\forall x$, $x_0 - \delta \leq x \leq x_0 + \delta$, $|f(x) - P(x - x_0)| \leq K|x - x_0|^{\alpha}$.*

Definition 2. *The Hölder exponent $h_f(x_0)$ of f at x_0 is the superior bound value of all α, $h_f(x_0) = sup\{\alpha, f \text{ is Hölder } \alpha \text{ at } x_0\}$.*

The local regularity of a function at a point x_0 is thus measured by the value $h_f(x_0)$. It is worth noting that the smaller $h_f(x_0)$ is, the more singular the signal is. For example, the Hölder exponent is 1 for a triangle function, 0 for a step function and −1 for a Dirac impulse.

To describe an ROI associated to an interest point in an image I_f, both orientation and Hölder regularity of singularities are characterized. The Hölder exponent is estimated in the gradient direction. For this purpose, orientation $\theta(x,y)$ and gradient magnitude $m(x,y)$ are computed at each pixel (x,y) :

$$\begin{cases} m(x,y)^2 = (I_f(x+1,y) - I_f(x-1,y))^2 \\ \qquad\qquad +(I_f(x,y+1) - I_f(x,y-1))^2 \\ \theta(x,y) = \tan^{-1}\left(\frac{I_f(x,y+1) - I_f(x,y-1)}{I_f(x+1,y) - I_f(x-1,y)}\right) \end{cases} \tag{1}$$

Then, for each singularity, the Hölder exponent α is estimated with foveal wavelets as presented in [9]. Orientations and Hölder exponents maps are then conjointly used and we approach their distribution with 3D histograms. To build such histograms, we consider a 32x32 ROI around each IP that we split into 16 8x8 subregions and we quantify the number of times each pair (α,θ) appears in each subregion (see Figure 2). We use three Hölder exponents bins into the range $[-1.5, 1.5]$ and eight orientation bins into $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. All bins of each subregion 3D histogram are concatenated to form the final signature : the Regularity Foveal Descriptor (RFD).

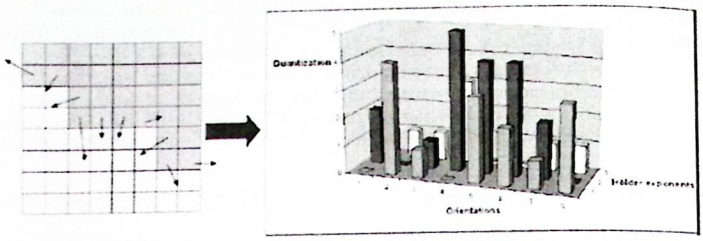


Fig. 2. Orientations and Hölder exponents for a subregion, resulting in a 3D histogram.

2.3. ASSOM Learning Process

ASSOM is basically a combination of a subspace method and a competitive selection and cooperative learning as in the traditional SOM, introduced by Kohonen [10]. ASSOM differs from other subspace methods by permitting to generate a set of topologically-ordered subspaces. Two units that are close in the map will represent two feature subspaces closed in the global feature space. In ASSOM, the unit is composed of several basic vectors that expand together a linear subspace. This unit is called "module" in an ASSOM neural network. This method aims at learning data features, without assuming any prior mathematical forms of their representation, such as Gabor or wavelet transforms, which are frequently encountered in the traditional image analysis and pattern recognition techniques. In other words, the forms of the filter functions are learned directly from the data. The input to ASSOM is a group of vectors, called "episode". The vectors in each episode are supposed to be close according to some affine transformation variations.

There are mainly two phases in the learning process with ASSOM:

1. For an input episode, locate the winning subspace from ASSOM modules.
2. Adjust the winning subspace and its neighbor modules in order to better represent the input episode.

For a linear subspace $\mathcal{L}$ of dimensionality H, one can find a set of basis vectors $\{b_1, b_2, \ldots, b_H\}$, such that every vector in $\mathcal{L}$ can be represented by a linear combination of these basis vectors. Such sets of basis vectors are not unique. However they are equivalent in the sense that they expand exactly the same subspace. For computational convenience, the basis vectors are orthonormalized by the Gram-Schmidt process.

The orthogonal projection of an arbitrary vector x onto the subspace $\mathcal{L}$, written as $x_{\mathcal{L}}$, is a linear combination of its orthogonal projections on the individual basis vectors, and can be computed by:

$$\hat{\mathbf{x}}_{\mathcal{L}} = \sum_{h=1}^{H} (\mathbf{x}^{\mathrm{T}} \mathbf{b}_h) \mathbf{b}_h. \tag{2}$$

If $\hat{\mathbf{x}}_{\mathcal{L}} = \mathbf{x}$, then x belongs to $\mathcal{L}$, otherwise we can define the distance from x to $\mathcal{L}$ as $\|\mathbf{x}_{\mathcal{L}_1}\| = \|\hat{\mathbf{x}}_{\mathcal{L}_2}\|$, by using the Euclidean norm. When several subspaces exist, the original space is partitioned into pattern zones and the decision surface between two subspaces, for example $\mathcal{L}_1$ and $\mathcal{L}_2$, is determined by those vectors x such that $\|\hat{\mathbf{x}}_{\mathcal{L}_1}\| = \|\hat{\mathbf{x}}_{\mathcal{L}_2}\|$. By comparing the distances of a vector to all the subspaces, we can assign this vector to the nearest subspace.

In Kohonen's realization of ASSOM, the subspace is represented by a twolayered neural architecture, as depicted in Figure 3. The neurons in the first layer compute the orthogonal projections $\mathbf{x}^{\mathrm{T}} \mathbf{b}_h$ of the input vector x on the individual basis vectors $\mathbf{b}_h$. The second layer is composed of a single quadratic neuron and computes the squared sum from the outputs of the first layer neurons.

The output of the whole neural module is then $\|\hat{\mathbf{x}}_{\mathcal{L}}\|^2$, the square of the norm of the projection. It can be regarded as a measure of the degree of matching of the input vector x with the subspace $\mathcal{L}$ represented by the neural module. In the case of an episode, the distances should be calculated from the subspace of the vectors in the episode and the subspace of the module, which are generally difficult to compute. Kohonen proposed another much easier but robust definition of subspace matching: the *energy*: the sum of squared projections over the episode on a module subspace. This is the energy that we use to build our activation histogram.

The classical Kohonen's ASSOM learning algorithm proceeds as follows.
For the learning step t,

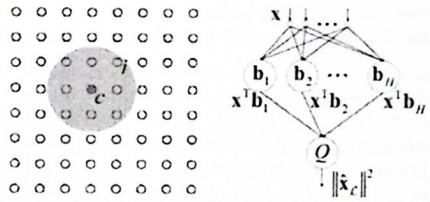


Fig. 3. Left: a rectangular ASSOM topology, a winning module c and its neighborhood. Right: the projection of x on $\mathcal{L}$ by a module.

1. Feed the input episode **x**, composed of S vectors **x**(s), $s \in S$. Locate the winning module indexed by c:

$$c = \arg\max_{i \in I} \sum_{s \in S} \|\mathbf{x}_{\mathcal{L}_i}(s)\|^2. \tag{3}$$

where I is the set of indices of the neural modules in the ASSOM.

2. For each module i in the neighborhood of c, including c itself, and for each input vector $\mathbf{x}(s)$, $s \in S$, adjust the subspace $\mathcal{L}_i$ by updating the basis vectors $\mathbf{b}_h^{(t)}$, according to the following procedure:
   (a) Rotate each basis vector according to:

$$\mathbf{b}_h^{(t)} = \mathbf{P}_c^{(t)}(\mathbf{x}, t)\mathbf{b}_h^{'(t)} \quad . \tag{4}$$

   In this updating rule, $\mathbf{b}_h^{(t)}$ is the new basis vector after rotation and $\mathbf{b}_h^{'(t)}$ the old one. $\mathbf{P}_c^{(t)}(\mathbf{x}, t)$ is the rotation operator matrix, defined as:

$$\mathbf{P}_c^{(t)}(\mathbf{x}, t) = \mathbf{I} + \lambda(t) h_c^{(t)}(t) \frac{\mathbf{x}(s)\mathbf{x}^{\mathrm{T}}(s)}{\|\dot{\mathbf{x}}_{\mathcal{L}_i}(s)\| \|\mathbf{x}(s)\|} \quad . \tag{5}$$

   where $\mathbf{I}$ is the identity matrix, $\lambda(t)$ a learning-rate factor that decreases with the learning step t. $h_h^{(t)}(t)$ is the neighborhood function defined on the ASSOM lattice with a support area shrinking with t.
   (b) Dissipate the components $b_{hj}^{(t)}$ of the basis vectors $\mathbf{b}_h^{(t)}$ to improve the stability of the results [10]:

$$\tilde{b}_{hj}^{(t)} = \mathrm{sgn}(b_{hj}^{(t)}) \max(0, |b_{hj}^{(t)}| - \varepsilon), \tag{6}$$

   where ε is the amount of dissipation, chosen proportional to the magnitude of the correction of the basis vectors.
   (c) Orthonormalize the basis vectors in module i.

2.4. Final Feature Vector Construction

The proposed architecture contains one ASSOM for each category, producing specific ASSOM units for different patches. This idea was explored in [11] for the recognition of handwritten digits and produced promising results. In this case, the image size was small (25 × 20 pixels) allowing a straightforward learning of all pixels through ASSOM. 10 ASSOMs were used, one trained for each category of handwritten digits. For digit classification, a test digit is sent simultaneously to all the 10 ASSOMs, which output 10 reconstruction error values. The ASSOM with the smallest reconstruction error determines the digit category. An obvious limitation here is that there is no interaction between the different ASSOMs during the learning phase. An ASSOM learns the features of its own category, however it does not learn how to separate them from the other categories. The optimum decision surface is thus not guaranteed.

In our context, the images to analyze are much larger and complex. Therefore, we decide to use a local approach by extracting image patches at salient locations. Our

strategy is thus to build a visual dictionary for each class from the activation of different ASSOMS. Other studies [12–16] are interesting regarding the creation of codebooks or bags of keypoints. Here, our object approach focuses on the pertinent image areas.

To construct the feature vector $\mathbf{H}_{\mathcal{I}}$ from the object $\mathcal{I}$ we proceed as follows (see Figure 1 for notations).

- We select the interest points with the strongest salience from the image $\mathcal{I}$
- For each patch:
  - We compute the local signature with the RFD descriptor (see 2.2).
  - We build an episode for the local signature by applying some artificial affine transformations.
  - For each episode vector:
    * J specialized ASSOM networks receive a signature and compute an energy $\|\hat{\mathbf{x}}_{kj}\|^2$ defined by:

$$\|\hat{\mathbf{x}}_{kj}\|^2 = \max_{i \in I_j} \|\hat{\mathbf{x}}_{k,i}\|^2 \quad . \tag{7}$$

where I_j is the module index set of the j^{th} ASSOM, J is the number of category. $\|\hat{\mathbf{x}}_{kj}\|^2$ is the maximal value of the square of the norm of the projection of $\mathbf{x}_k$ on the linear subspaces of the j^{th} ASSOM.

    * Each activation histogram h_j corresponding to each network is then updated. The maximal output energy increments the corresponding histogram bin, as follows.

$$h_j[i^*](t+1) = h_j[i^*](t) + \|\hat{\mathbf{x}}_{kj}\|^2 \tag{8}$$

with $i^* = \arg\max_{i \in I_j} \|\hat{\mathbf{x}}_{k,i}\|^2$ and t is the time.

- Each energy histogram h_j, computed from all patches, is fused into a global activation histogram $\mathbf{H}_{\mathcal{I}}$. This final feature vector is then the concatenation of the individual ASSOM energy histograms. This discriminative object information is finally introduced to a SVM classifier for supervised training.

2.5. Object Detection by multi-resolution

The visual concept learning process is tuned on normalized object samples. Thus, the classification procedure consists of an object search. This search is made by a fixed size sliding window in a multi-scaled image pyramid.

Here, the object detection is realized on three pyramid levels and the window moves with a step of the half of its width (see Figure 4).

For each image extracted inside the sliding window, we observe the SVM outputs. When the classifier recognize a learned object, the corresponding area is marked in a voting map. The vote weight is proportional to the SVM output given that the output i

of the SVM classifier represents the *a posteriori* probability of an object of belonging to class *Ci*.

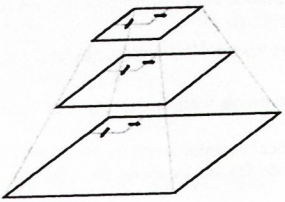


Fig. 4. Scaling pyramid and its moving window.

A last step fuses the different vote intensities in order to locate the object. This procedure clusters the multi-scale vote intensities in the original dimension (see Figure 5). Thus, we locate some candidate areas where the sum of multiscale votes is upper than a decision rule threshold. Typically, when one of the SVM outputs is greater than 0.9, we consider the corresponding area as relevant. Then, a final classification refines the results in the merged areas.

3. Experimental results

We tested the proposed scheme on the challenge PASCAL 20051 database. The goal is to classify 689 images using 684 training labeled objects of four different classes: bicycle, car, motorbike and people.

The local signatures are extracted around the IPs within 32x32 rectangular patches. The RFD descriptor is computed within 16 subregions of these patches, 8 orientation bins and 3 Hölder exponent bins for all patches. Therefore, the RFD dimension is 16x8x3=384.

For all experiments, we configure our ASSOM networks with the following rules for optimal performance in terms of accurate data representation:

– the number of training epochs is : $T = 500 \times N$;

– the learning rate forms a monotonically decreasing sequence: $\lambda(t) = \dfrac{T}{T + 99t}\lambda(t)$:

– the neighborhood function is defined by: $h(i) = \begin{cases} 1, r_c - r_i < \mu(t) \\ 0, otherwise. \end{cases}$

Here, we choose the euclidean norm and r_i is the 2D position of the i^{th} ASSOM module. $\mu(t)$ specifies the neighborhood width which decreases linearly with t from $\dfrac{\sqrt{2}}{2}N$ to 0.5.

The *Area Under Curve* (AUC) and the confusion matrix of the image classification system with the best configuration are shown in Table 1. The best-classified category is "cars": 92.39% are correctly detected. The worst case happened to be the people category. This can be explained by the large variety of textures, colors and shapes in this cluster.

Table 1. Confusion matrix and AUCs. (B=Bicycle, C=Car, M=Motorbike, P=People)

Classified as →	B	C	M	P	Classes	AUC
B	82	12	14	6	B	0.863
C	2	243	5	13	C	0.938
M	3	11	199	3	M	0.944
P	9	17	6	52	P	0.879



Fig. 5. Good and false classifications. Voting maps for a bicycle image.

Some correctly classified examples are shown in Figure 5. We can observe that a bicycle object stimulates strongly the bicycle-voting map. Moreover, the motorbike-voting map is lightly activate, which demonstrates the generalization power of the proposed system.

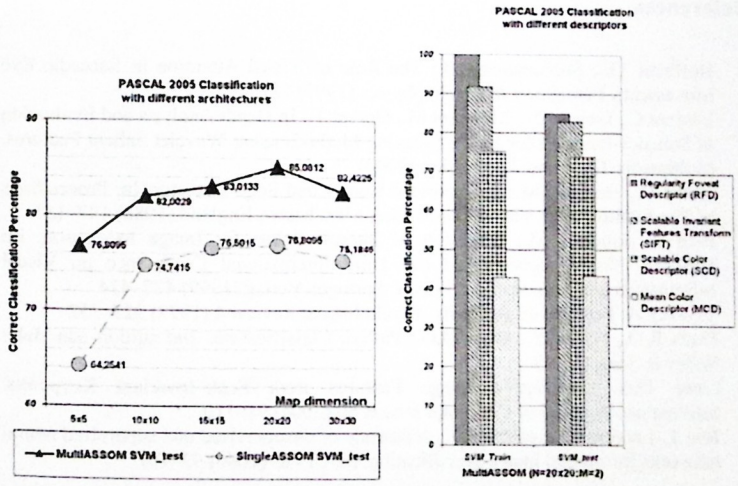


Fig. 6. Results on PASCAL 2005 database with architecture and descriptor variations.

This multi-ASSOM architecture offers a better global classification rate (85.08%) than a single ASSOM (76. 81%) for all test classes. The best configuration for an ASSOM network is : N=20x20. M=2 (see Figure 6). It is worth noting that the global classification rate for the training database reaches 100% for our multi-ASSOM scheme. and only 89.96% for the single ASSOM scheme.

It is also interesting to compare the performances when using different descriptors. With the same configuration. the RFD descriptor provides better results than the SIFT descriptor or some MPEG-7 descriptors. Consequently, we can see that the ASSOM competition with the RFD descriptor allows us to construct more discriminative feature vectors for the SVM classification[2].

4. Conclusion

This article describes a new system to detect visual concepts. using singularity information contained in the salient regions of interest. Based on the three main properties of ASSOM - which are dimension reduction, topology preservation and invariant feature emergence - our scheme give very promising results to detect objects with a SVM classifier.

We plan to study the fusion of heterogeneous descriptors with feature selection to learn useful object information. and to develop a growing strategy to find the optimal ASSOM parameters.

References

1. Hoffman J.E., Subramaniam B.: The Role of Visual Attention in Saccadic Eye Movements. Perception & Psychophysics (1995) 787–795
2. Laurent C., Laurent N., Maurizot M., Dorval T.: In Depth Analysis and Evaluation of Saliency-based Color Image Indexing Methods using Wavelet Salient Features. Multimedia Tools and Application (2004)
3. Harris C., Stephens M.: A Combined Corner and Edge Detector. In: Proceedings of The Fourth Alvey Vision Conference. Manchester, England (1988) 147–151
4. Bres S., Jolion J.-M.: Detection of Interest Points for Image Indexation. In: VISUAL '99: Proceedings of the Third International Conference on Visual Information and Information Systems. Springer-Verlag (1999) 427–434
5. Tversky A: Features of similarity. Psychological Review 4 (1977) 327–352
6. Duda R.O., Hart P.E., Stork D.G.: Pattern Classification. 2*nd* edition edn. John Wiley & Sons (2001)
7. Lowe D.G.: Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60 (2004) 91–110
8. Ros J., Laurent C., Lefebvre G.: A cascade of unsupervised and supervised neural networks for natural image classification. In: CIVR. (2006) 92–101

[2] WEKA SVM classification (http://www.cs.waikato.ac.nz)

9. Mallat S.: A Theory for Multiresolution Signal Decomposition: The Wavelet Representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 11 (1989) 674–693
10. Kohonen T.: Self-Organizing Maps. Springer (2001)
11. Zhang B., Fu M., Yan H., Jabri M.A: Handwritten digit recognition by adaptivesubspace self-organizing map (ASSOM). IEEE Transactions on Neural Networks 4 (1999) 939–945
12. Csurka G., Bray C., Dance C., Fan L.: Visual Categorization with Bags of Keypoints. In: The 8th European Conference on Computer Vision. Prague, Czech Republic (2004) 327–334
13. Quelhas P., Monay F., Odobez J.-M., Gatica-Perez D., Tuytelaars T., Van Gool L.: Modeling scenes with local descriptors and latent aspects. In: IEEE Int. Conf. on Computer Vision. (2005) IDIAP-RR 04-79.
14. Fei-Fei L., Fergus R., Perona P.: One-shot learning of object categories. IEEE Transactions on PAMI 4 (2006) 594–611
15. Lazebnik S., Schmid C., Ponce J.: Spatial pyramid matching for recognizing natural scene categories. IEEE CVPR 2 (2006) 2169–2178
16. Lefebvre G., Laurent C., Ros J., Garcia C.: Supervised image classification by som activity map comparison. In: Pattern Recognition, 2006. ICPR 2006. 18th International Conference on. Volume 2. (2006) 728– 731.

Data Analysis, Pattern Recognition and Self-Organization

Analysis and Classication of Car Engine Sounds to Diagnose Failures with the Help of Neural Networks

Luis Andrés Méndez-Ortega[1] and Orion Fausto Reyes-Galaviz[2]
[1]Universidad Autonoma de Tlaxcala
Facultad de Ciencias Básicas, Ingenieria y Tecnologia
http://www.ingenieria.uatx.mx/licenciaturas/ic/
[2]Instituto Tecnologico de Apizaco
http://www.itapizaco.edu.mx/
Apizaco, Tlaxcala. México
E-mail: [1]siul222@hotmail.com, [2]orionfrg@ingenieria.uatx.mx
(Paper received on August 12, 2006, accepted on September 25, 2006)

Abstract. The detection of possible car engine failures can be achieved through the identification on the vibration changes that are produced while the motor is operating. A person with enough mechanical experience is able to recognize a flaw in the engine, just by listening to the noise and sensing the vibrations produced. A car owner can notice a flaw in the motor, when a shift on the movement of the engine is felt, or if there's an abnormal noise on it. Nevertheless, it is hard to guess what kind of malfunctions produces certain motors noises and vibrations, making it hard to work directly on the engine's flaw. In this work, a method is proposed to classify, and directly detect, some common car engine failures, just by using the engine's recorded sound. To accomplish this goal, we used Linear Prediction Cepstral Coeficients (LPCC) as the main acoustic characteristics extractor, Principal Component Analysis, and a Time Delay Neural Network for the pattern recognition. We obtained recognition results of up to 95.58% when predicting four common car failures.

1. Introduction

A failure on any engine can produce mild or violent vibrations that can damage the motor; if left ignored. These vibrations can cause fissures, waste and/or overheating of important parts and reduce the machine's performance. Besides, the vibrations are a good indicator of the mechanical performance and they are also very sensitive to the malfunction evolution. In general, these types of flaws are preceded by shifts on the sound conditions, vibration, power loss, etc. These indicators are a sign of some kind of future failure on the engine's performance. Although, not all the sources of sounds or vibrations are inevitable, due to the fact that some are natural to the machine's operation by it self, enhancing with this the importance of identifying which correspond to a possible malfunction. Most of the sounds that an engine produces are generated by rotating mechanisms inside the motor; also, the machines that have an internal combustion system, produce more noise due to the explosions created inside.

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 75-85

In general, a trained ear can quickly detect the existing differences between the noises that are called normal, and a series of abnormal noises produced, for example, by vibrating gears, poundings on the engine, hisses, etc. If this kind of relevant acoustic information exists, hidden inside the engine's noise, the extraction, recognition, and classification of the acoustic characteristics could be possible, using automatic mechanisms, in order to obtain reliable diagnostics.

The engine noise can be an important clue to the prediction of possible failures, and can be helpful to preserve a good motor performance. The analysis presented in this work, joint with the monitoring systems that actually exist, can produce more tools that can help the motor specialists obtain robust diagnostics when repairing car engines. Taking this into account, it is proposed to capture and analyze the acoustic sound produced by these kind of machinery, and use techniques of acoustic characteristics extraction, and pattern recognition, in order to classify the different types of noises that are produced by engines with good and bad performance. On the next section we will present a review of prior comparative studies on the field. Section 3 details the fundamental basis in the noise recognition process and describes our proposed system. Section 4 deals with acoustic processing and the feature extraction method which uses the Linear Prediction Cepstral Coefficients (LPCCs) method. A fundamental theory on pattern classification, Principal Component Analysis, and Time Delay Feed-Forward Neural Networks, is given in Section 5. The complete system implementation can be found on Section 6. Our experimental results, that go up to 95.58%, are shown in section 7, and the concluding thoughts are presented in Section 8.

2. State of the Art

Many systems have been proposed on the acoustic pattern recognition field in the past few years. Depending on the problem that has to be solved, a common point of some approaches is focused on the detailed spectral analysis of the acoustic signal. In [1] Mario E. Munich uses Mel Frequency Cepstral Coefficients (MFCC) for the acoustic analysis, Gaussian Mixture Models (GMM), and Hidden Markov Models, combined with Bayesian Subspace Methods, applied to the automatic recognition of acoustic characteristics from vehicles; this is used on military operations, for surveillance purposes, achieving a precision of 83%. Huadong Wu, Et Al [2], also worked on the vehicle recognition field. They proposed that each vehicle model has the same kinds of noises, vibrations, hops, and tire friction. They used a method called eigenfaces, used most commonly on the face recognition field, to characterize the noise patterns and use them to recognize the vehicle; this method is also known as the Korhunen-Loeve expansion or as the Principal Component Analysis (PCA), the results on this work seem promising. In [3], Edgar A. Estupiñan, Et Al use an analysis of mechanical vibrations, as a part of their predictive maintenance, to establish the mechanical health of their machines, preventing with this, future flaws. They propose techniques based on the Fast Fourier Transform to analyze the noise produced by the vibration of slow rotatory machinery. Pedro N. Saavedra [3], [4], presents some techniques applied to the vibration analysis; he particularly analyzed the use of fissure detection on machinery

axis and rafters. For the theoretical study, the Finite Element was used, and the fissure was modeled using lineal fractomechanical theory.

It is difficult to compare these systems; because the data bases, experimental conditions, number of samples per second, frame size, and type of recognitions, are different. Although, the analysis of different techniques, suggests interesting possibilities of combining them in order to obtain better recognition results.

3. Car Engine Noise Recognition Process

The car engine noise recognition process is basically a pattern recognition problem, and it is similar to speech recognition. The goal is to take the motor's sound wave as an input, and at the end recognize the engine's malfunction. Generally, the engine noise recognition process is done in two steps; the first step is the acoustic processing, or features extraction, while the second is known as pattern processing or classification. The proposed system can be seen in Figure 1. For this case, in the acoustic analysis, the engine's signal is processed to extract relevant features in function of time. The feature set obtained from each noise sample is represented by a vector, and each vector is taken as a pattern. As for the pattern recognition methods, four main approaches have been traditionally used: pattern comparison, statistical models, knowledge based systems, and connectionist models. We focus on the use of the last one.

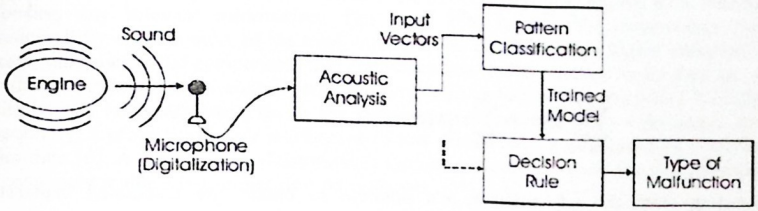


Fig. 1 Engine Noise Recognition Process.

4. Acoustic Processing

The acoustic analysis implies the application and selection of filter techniques, feature extraction, signal segmentation, and normalization. With the application of these techniques the signal is described in terms of its fundamental components. An engine signal is too complex and codifies more information than the one needed to be analyzed and processed in real time applications. For this reason, in the engine noise recognition process we use a feature extraction function as the first plane processor. Its input is the engine sound signal, and its output is a vector of features that characterizes

key elements of the sound wave. In this work we used Linear Prediction Cepstral Coefficients (LPCC) [5] as the feature extraction method.

4.1. Linear Prediction Cepstral Coefficients

The Lineal Prediction (LP) method is historically one of the most important methods used for the voice analysis [5]. Its fundamental basis is to establish a filter model for the sound source. With enough number of parameters, the LP model can establish a suitable approximation to the spectral structure of any kind of sound. This is why this technique is used to analyze our acoustic samples. The LP method gets this name because it pretends to extrapolate the value of the sound's next sample $x(n)$ as the weighted sum of the previous samples $x(n-1), x(n-2), ..., x(n-k)$:

$$x(n) = \sum_{i=1}^{K} a_i x(n-1) \tag{1}$$

To do so, a coefficient computation must be made, by minimizing an error function E, specifically the least square, over a window of size N.

$$E = \sum_{n=0}^{N-1} e^2(n) \tag{2}$$

$$E = \sum_{n=0}^{N-1} \left(x(n) - \sum_{i=1}^{K} a_i x(n-1) \right)^2 \qquad 0 \le n \le N-1$$

Departing from the LP analysis, it's possible to obtain the associated cepstral coefficients expression (LPCC):

$$c(0) = \log(1) = 0 \tag{3}$$

$$c(i) = -a(i) - \sum_{j=1}^{i-1} \left(1 - \frac{j}{i}\right) a(j) c(i-j) \quad 1 \le i \le N_c \tag{4}$$

A common transformation, over this kind of coefficients, is known as the delta cepstral coefficients or delta cepstrum coefficients. We can obtain these by applying the next expression:

$$\Delta c_j(i) = \frac{1}{2T+1} \sum_{k=-T}^{T} k \cdot c_{j+k}(i) \tag{5}$$

5. Engine Noise Pattern Classification

After extracting the acoustic features of each noise sample, the feature vectors are obtained; each one of these vectors represents a pattern. These vectors are later used for the classification process. For the present work, we focused on connectionist models, also known as neural networks (NN), to classify these vectors (patterns). The NN are combined with Principal Component Analysis to reduce the vector's dimensionality; in order to improve the training/testing processing time, obtaining with this, a more efficient classification system.

5.1. Principal Component Analysis

The Principal Component Analysis (PCA), is commonly used in signal processing, statistics, and neural computation. The objective of PCA is to reduce the dimensionality of a variable set p to a set m with less variables to improve the readability of the data. The variable set, or original characteristics, occasionally are redundant, that is why it's useful to reduce the dimension of the original data, without loosing any relevant information. The PCA tries to find the components that successfully explain most of the total variation; the data that has a higher variation is kept, eliminating the components that contribute less to the variation in the data set. A reduced set is easier to analyze and interpret. This reduction has important benefits; firstly, the computational cost in the subsequent processing stages is reduced, and secondly, a projection inside a subspace of low dimensions is very useful to visualize the data [6]. A simple PCA illustration is shown in Figure 2, where the first principal component inside a two-dimension set is shown.



Fig. 1. Principal Component Analysis of a bidimensional data cloud. The line shows the direction of the first principal component.

5.2. Neuronal Networks

Artificial neural networks (ANN) are widely used on pattern classification tasks, showing good performance and high accuracy results. In general, an artificial neural network is represented by a set of nodes and connections (weights). The nodes are a simple representation of a natural neural network while the connections represent the data flow between the neurons. These weights are dynamically updated during the network's training. In this work, we will use the Feed Forward Time Delay Neural Network model; because this model has shown good results in voice [7] and non-voice [8] recognition tasks.

5.3. Feed-Forward Time Delay Neuronal Network

Input Delay refers to a delay in time, in other words, if we delay the input signal by one time unit and let the neural network receive both the original and the delayed signals, we have a simple time delay neural network. This neural network was developed to classify phonemes in 1987 by Weibel and Hanazawa [9].

The Feed-Forward Time Delay neural network doesn't fluctuate with changes; the features inside the sound signal can be detected no matter in which position they are in. The time delays let the neural network find a temporal relation directly in the input signal and in a more abstract representation of the hidden layer. It does this by using the same weights for each step in time [7].

5.4. Scaled conjugate gradient back-propagation

The neural network training can be done through a technique known as Backpropagation. The scaled conjugate methods are based on the general optimization strategy. We use scaled conjugate gradient back-propagation (SCGBP) to train the neural networks; because this algorithm shows a lineal convergence on most of the problems. It uses a mechanism to decide how far it will go on a specific direction, and avoids the time consumption on the linear search by a learning iteration, making it a fast second order algorithm [6].

6. System Implementation

In most of the towns and cities in the Mexican territory, the use of public transportation is very common. One car model, that is commonly used for this purpose, is the *Combi Van* from Volkswagen, models '82, '83, '86, '88, and '90.

These cars work almost every day from 6:00 a.m. to 21:00, and there are some hours that they carry up to 18 persons and travel several kilometers with this weight. Some car shops, in cities where these vehicles are commonly used as public transportation, specialize on maintaining and repairing only these kind of engines (Figure 3), giving the engine specialists, the ability to quickly diagnose the most common motor flaws, and directly work on the problem of these specific engines. For

these reasons we choose this kind of engines. The conditions to select and record an engine were:

- Old models or engines previously repaired.
- Similar engine characteristics between engines.
- Several similar car models, or with the same engine.

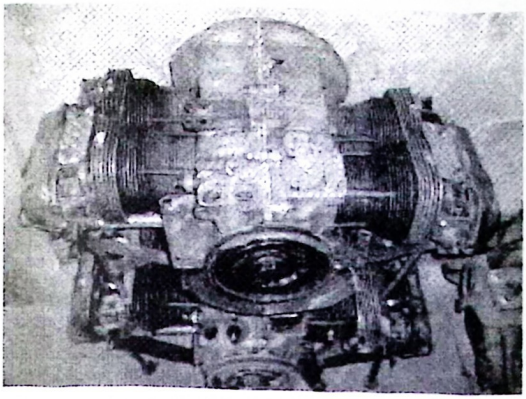


Figure. 2. Volkswagen *Combi* engine with 4 cylinders.

We recorded the engine sound in two phases; when the car first arrived to the shop and after the car was diagnosed and repaired. This method helped us label the recordings and compare the noise produced by the engine before and after the service. The two phases consisted where; accelerating the engine, and in *Ralenti* state (minimum speed). Also, some noises were recorded from tuned engines in both phases.

6.1. Noise Samples

To record the engine's noise, we visited car shops recommended by public transportation drivers. The recorded samples were obtained at five different car shops, three of them specialized only on Combi Vans. Each recorded sample was obtained with the help of a ZicPlay digital recorder, and the output files were saved in WAV format with the following settings; 8 Khz, 16 bits, and monaural. Each of these files has a time period of 3 to 5 seconds, and we recorded a total of 26 different engines. These samples were classified in four categories, tuned engines (Class 1), crank related problems (Class 2), piston related problems (Class 3), and valves related problems (Class 4), see Table 1. On Table 2 we can see the number of samples obtained from engines on Ralenti state.

Table 1. Recorded samples from accelerating engines, per class.

Identified Problem	Number of Samples	Time (in seconds)
Class 1	7	18.036
Class 2	6	18.502
Class 3	6	17.664
Class 4	7	18.006
Total	26	72.208

Table 2. Recorder samples from engines in Ralenti state, per class.

Identified Problem	Number of Samples	Time (in seconds)
Class 1	7	16.396
Class 2	6	15.430
Class 3	6	15.744
Class 4	7	16.590
Total	26	64.208

It has to be taken into account that these samples were difficult to obtain; in some cases some samples had to be rejected, due to the poor sound quality, minor flaws, and other issues that made the sample unusable for the system's goals. 3 different flaws were used, plus one from tuned/fixed motors, because according to the experts, these were the most common flaws. At first, five common flaws were considered, but two of them directly related to the three flaws used for this work, ending with only 3.

The sound recordings, obtained from engines in both, accelerating and Ralenti states, where segmented in 1 second samples and labeled from classes 1 thru 4; for each segmented sample we extracted a different LPCCs number (as shown on Tables 3 and 4). On one experiment, the vectors were used without any reduction, and in another experiment, the resulting vectors were reduced to 52% its original size by using PCA. The LPCC configurations and PCA reduction were obtained heuristically, and were the ones that gave the best overall results. The reduced vectors are then used to train/test the TDNN; we used 80% of the segmented samples to train the NN and 20% to test it. The network architecture consists on; an input layer with the number of nodes corresponding to the input vectors size, a hidden layer with 60% less nodes than the input layer, and an output layer corresponding to the number of classes to predict, in this case 4. The NN was implemented with the help of the Neural Network Tool-box, in Matlab v7.0, and the characteristic extraction was done with the help of the Auditory Tool-box for Matlab [10].

7. Experimental Results

On Table 3 we can see the results obtained by using the samples from engines in Ralenti state. Table 4 shows the results obtained from samples recorded from

accelerating engines. Each result is the best overall result from 10 experiments. On both tables, the first column shows the number of characteristics extracted (c) per each time frame (ms).

Table 3. Results obtained with samples from engines in ralenti state.

Configuration	Precision (without PCA)	Precision (without PCA)
LPCC 36c 100ms	95.22%	94.37%
LPCC 26c 100ms	95.41%	91.62%
LPCC 36c 100ms	89.74%	78.55%

Table 4. Results obtained with samples from accelerating engines.

Configuration	Precision (without PCA)	Precision (without PCA)
LPCC 36c 100ms	95.55%	94.27%
LPCC 26c 100ms	93.01%	95.58%
LPCC 36c 100ms	86.47%	82.58%

The results on both tables show that the best precision was given when using 36 and 26 LPCCs for each 100ms time frame, when using the samples from accelerating engines; here we can say with 36 coefficients the training time was longer, and there is not so much difference on the recognition results, showing with this that 26 LPCCs give the best overall results. Taking this into account, we randomly selected 12 WAV files from the training samples, previously separated, and fed them to the trained NN, calculating with this the precision percentage given by each output node. The results obtained are shown in Table 5.

8. Conclusions and Future Work

The results show that a motor in bad conditions does in fact emit different sounds than a tuned motor, also that the sounds are different, or at least are recognized by the TDNN, depending on the type of flaw that's affecting the motor efficiency. This information can help the motor specialist give fast diagnosis, prices estimates for the reparation, and work directly on the motor's problem. We also concluded that is viable to work on this problem and build a cheap system that can be used by motor manufacturers and car shop owners.

We are working with other acoustic characteristic extraction such as MFCC and LPC, and noise reductions methods, by using wavelets filters, in order to compare the shown results with these methods, and make a trustworthy system. We are also collecting more sound samples from Combi Van motors and from other car brands, to build a system that is able to recognize other motors and other failures. For the final tests on Table 5 we worked on a user interface, designed in Matlab, that accepts WAV files as an input, automatically extracts the acoustic features and, with a trained NN

previously loaded, emits the predicted failure. But, we want to design an independent user interface that can be used on car shops and predict engines failures, only by feeding directly the motor noise, by microphone, or pre recorded engine noise samples.

Table 5. Comparison between the predicted class and the real class.

Test	Class 1	Class 2	Class 3	Class 4	Real Class
1	94.5945	0.0	5.4054	0.0	yes
2	100.0	0.0	0.0	0.0	yes
3	89.1891	0.0	0.0	10.8108	yes
4	0.0	75.6756	21.6216	2.7027	yes
5	5.4054	94.5945	0.0	0.0	yes
6	2.7027	67.5675	8.1081	21.6216	yes
7	0.0	0.0	97.2972	2.7027	yes
8	0.0	5.4054	86.4864	8.1081	yes
9	0.0	0.0	2.7027	97.2972	no
10	0.0	0.0	0.0	100.0	yes
11	0.0	0.0	8.1081	91.8918	yes
12	0.0	0.0	8.0181	91.891892	yes

References

[1] Mario E. Munich. Bayesian Subspace Methods for Acoustic Signature Recognition of Vehicles. Published at the 12*th* European Signal Processing Conference (EUSIPCO-2004), Vienna, Austria. September 6-10, 2004.

[2] Huadong Wu, Mel Siegel, and Pradeep Khosla. Vehicle Sound Signature Recognition by Frequency Principle Component Analysis. The Proceedings of IMTC'1998, selected in the IEEE Transaction on Instrumentation and Measurement Vol. 48, No. 5, page 1005 - 1009, (ISSN 0018-9456), October 1999.

[3] Pedro N. Saavedra and Edgar A. Estupiñan. Análisis De Vibraciones Aplicado A Las Máquinas Rotatorias De Baja Velocidad. http://internal.dstm.com.ar/sites/mmnew/bib/notas/Estupi1.pdf. Chile 2000.

[4] Pedro Nelson Saavedra. La Medición y Análisis de las Vibraciones como Técnica de Inspección de Equipos Y Componentes, Aplicaciones, Normativas Y Certificación. Facultad de Ingeniería - Universidad de Concepción, Casilla 160 - Concepción -Chile. http://www.aaende.org.ar/sitio/biblioteca/material/CONFCHILE.pdf. 2001.

[5] Agustín Álvarez Marquina. Fundamentos del Reconocimiento Automático de la Voz, Algoritmos de Extracción de Características. Facultad de Informática. Universidad Politécnica de Madrid (UPM). November 22, 2004.

[6] Orozco-Garca, Jos. Extracción y Análisis de Características Acústicas del Llanto de Bebés para su Reconocimiento Automático Basado en Redes Neuronales. Tesis. INAOE, México. February 2004.

[7] Hilera J., Martínez V. Redes Neuronales Artificiales Fundamentos, modelos y aplicaciones. Alfaomega, Madrid, España. (2000).

[8] Reyes-Galaviz O. F., Reyes-García, C.A., Infant Cry Classification to Identify Hypoacoustics and Asphyxia with Neural Networks, in LNCS 2972: Advances in Artificial Intelligence. Springer, Berlin, 2004, pp 69-78, ISBN 3-540-21459-3, ISSN 0302-9743.

[9] A. Weibel, T. Hanazawa, G. Hinton, K. Shikano, and K.J. Lang, "Phoneme Recognition Using Time Delay Neural Networks," IEEE Trans. Acoustics, Speech, Signal Proc., ASSP-37: 32'339, (1989).

[10] Auditory Toolbox. Malcolm Slaney, Interval Research Corporation http://rvl4.ecn.purdue.edu/ malcolm/interval/1998-010/AuditoryToolbox.zip

Personalizing Google's Search Results using Naïve Bayesian Probabilistic Model

SRINIVASA K G, SRINIVAS KUMAR M, VINAY T C
Data Mining Laboratory, M S Ramaiah Institute of Technology,
Bangalore *560054*, India
kgsrinivas@msrit.edu, {skumar.msrit, vinaytc}@*gmail.com*

VENUGOPAL K R
Department of Computer Science and Engineering
University Visvesvaraya College of Engineering, Bangalore University
Bangalore *560001*, India
venugopalkr@gmail.com

L M PATNAIK
Microprocessor Applications Laboratory
Indian Institute of Science, Bangalore, India
lalit@micro.iisc.ernet.in

(Paper received on August 24, 2006, accepted on September 27, 2006)

Abstract. Personalization is the process of presenting the right information to the right user at the right moment, by storing browsing history about the user and analysing that information. Personalization methods enables the site to target advertising, promote products, personalize news feeds, recommend documents, make appropriate advice, and target e-mail. In this paper we introduce a Naive Bayesian probabilistic model, which classifies the sites into different categories. The user profile is built dynamically on the recorded interests of the user, which are nothing, categories of the site in which user browses. The algorithms are tested on varying keywords and the results obtained were compared with the Google's page rank system.

1. Introduction

Internet is one development, which fuelled the growth of IT industry to a large extent. It has single handedly changed the way we communicate, collaborate and cooperate with each other. It has been responsible for breaking the barriers of time and geographical limitations and helped set a new business order. From being an information repository it has today grown to be a strategic tool in business. Internet, by its very nature, does not support systematic storage of information. The units of storing information in Internet servers are websites that are uniquely identified by their URL. The problem that everyone had to face aftermath the explosion of information in Internet was to get relevant information on time. Remembering URLs proved cumbersome and difficulty in mining for correct information was proving as hindrance for the further growth of Internet.

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 87-96

Search Engines proved to the right medicine for this ailing problem of the industry. But search engines faced a typical problem since they relied on keywords for conducting search. The present day search engines have failed to understand the specific requirements of the user [17]. Users experience increasing difficulty finding documents relevant to their interests as search engines throw same result to everyone irrespective of user expectation. Google's Personalized Search drives to make search experience more relevant to the user. Using Personalized Search, user can get the results most relevant to user, based on what user has searched in the past, View and manage user's past searches, including the web pages, images, Froogle results which he has clicked on and create bookmarks. Personalized Search orders your search results based on user's past searches, as well as the search results and news headlines user has clicked on. User can view all these items in your Search History and remove any items you'd like. Early on, user may not notice a huge impact on his search results, but as he builds up your search history, his personalized search results will continue to improve.

2. Related Work

Conceptual search can be done by explicitly providing the meaning of the content in a Web page. So one way to address this problem is by having the authors of the content explicitly specify the meaning associated with a page using a Knowledge Representation Language. One of the Knowledge Representation Languages is *Ontobroker* and is discussed in [1]. Domain-specific Web search engines are effective tools for reducing the difficulty experienced when acquiring information from the Web. Building of a domain-specific search engine simply by adding domain-specific keyword, called "keyword spices," to the user's input query and forwarding it to a general-purpose Web search engine is presented in [2]. A tool that assists an end-user or an application to search and process information from the Web pages automatically by separating the primary content sections from the other content sections is presented in [3]. Ontology is a specification of a conceptualisation. Sophisticated ontologies incorporate logical relationships and membership rules. However, concept hierarchies can also be used as simple ontologies. Use of Yahoo! categories as a concept hierarchy and classifying documents into it using an *n-gram* classifier is discussed in [4].

he user profiles are a representation of the user's interests, such as Wisconsin Adaptive Web Assistant (WAWA). Building profiles non-invasively by observing user's visit to Web pages over a period of time is addressed in [5]. They generally use the profile to suggest related Web pages to the users as they browse. The study of personalized recommendation in a B2C portal to build improved algorithm, EI-B&B-MDL, for learning Bayesian networks effectively and efficiently is proposed in [6]. The algorithm reduces the number of independence tests and database passes while effectively restricting the search space. A personalized recommendation agent, fuzzy cognitive agent, to give personalized suggestions based on the current user's preferences, general user's common preferences, and the expert's knowledge is given in [7]. Fuzzy cognitive agents are able to represent knowledge via extended fuzzy cognitive maps, to learn user's common preferences from most recent cases and to help customers to make in-

ference/decisions through numeric computation instead of symbolic and logic deduction. An algorithm has been presented in [8] which can generate and display helpful links while users navigate a site and hence increasing the Web site's usability and help Web designers and the user achieve their goals.

In the literature data mining methods are very much exploited to build the customer profiles [9]. The 1:1 Pro system constructs profiles based on customers' transactional histories. The system uses data mining techniques to discover a set of rules describing customers' behaviour and supports human experts in validating the rules. The vision of ontology learning including a number of complementary disciplines that feed on different types of unstructured, semi structured and fully structured data to support semiautomatic, cooperative ontology engineering is presented in [10]. A new method for tracking the dynamics of user interests from a minimal number of relevance judgments is given in [11].

3. Proposed System

Problem Formulation: Search engines are affected by problems such as ambiguity [16] and results ordered by Web site's popularity rather than user interests. Natural language queries are inherently ambiguous. For example, consider a user query "Pluto". Due to ambiguity in the query terms, the results obtained are either related to astronomy or cartoon. Most users enter quires in just a word or two without providing enough information. These short queries are often ambiguous, providing little information to the search engine. A user profile that represents the interests of a specific user can be used to supplement queries, narrowing down the number of topics considered when retrieving the results. If system had a prior knowledge that user has a strong interest in Astronomy and little in others like cartoon, the Astronomy related results of Pluto could be presented to the user first and then cartoons preferentially. Therefore, user profile creation is important for personalization [18].

Our approach of building user profiles is based on the user's interactions with a particular search engine. For this purpose, GoogleWrapper: a wrapper around the Google search engine [13] is implemented, it logs the queries, search results, and clicks on a per user basis. The snippets, which are obtained using wrapper serves as the input for the algorithm to identify the category of that site. This information is then used to create user profiles and these profiles are used in a controlled study to determine their effectiveness for providing personalized search results. Information about the user can be collected either explicitly or implicitly. Explicit construction is the way to fill up the forms at the time of first login giving preferences or ratings. Implicit construction is observing user behaviours such as the categories of the URLs visited, time spent and number of inner clicks made in a particular site. Explicit construction of user profiles has several drawbacks. The user may provide inconsistent or incorrect information and the profile built is static whereas the user's interests may change over time, and the construction of the profile places a burden on the user. User browsing histories i.e., the categories the user has browsed so far are the most frequently used source of information about user interests. This information is used to create user profiles. Classifying the collected Web pages with respect to a category in which the user is interested cre-

ates the user profile. The fact that a user has visited a page and spent some time is an indication of user interest in that page's content i.e., in that category.

Problem Definition: Consider a Search Engine E, which accepts queries in the form of keywords and returns a list of near relevant web pages. The queries are of the form k_1 [**(op)** k_i] for i= 2 to n

Where k_i is the i^{th} keyword supplied to E

(op) is a Boolean operator like OR, AND...

n is the number of keywords supplied to E

[...] indicated the parameters are optional.

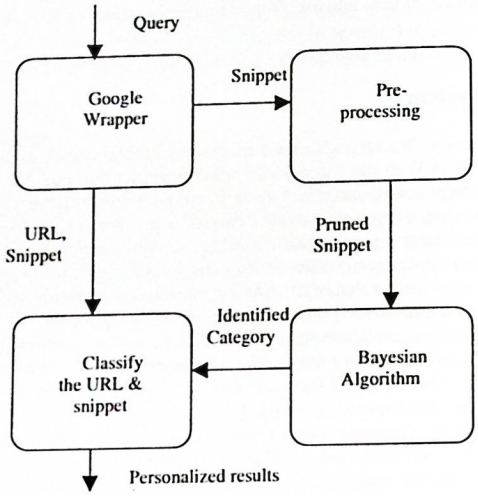


Fig. 1. . System Architecture.

When search engine E returns its results, user clicks on some site and then the corresponding snippet S is used as an input to the algorithm to identify the category of the site. This category represents the user's interest and is added to the user profile. Each category has an associated weight and the weight decides the choice of the category. Each time a new category is created, it is assigned an initial weight $C_{Weight} = 0$. Hence all categories start with the same weight or same opportunity in the user profile. The weights for a category are decided dynamically to reflect the user's interest in certain domain. The weight of the category also depends on the amount of time the user spends in a particular site and therefore in a particular category. A higher weight indicates a greater interest in that domain. When the weight for a category crosses a pre-defined

lower threshold, it is deleted from the user profile. This reflects the fact that the user has lost interest in that domain. The overall system architecture is given in Figure 1.

Table 1: Naïve Bayesian Probabilistic Model

Algorithm: Naïve Bayesian Probabilistic Model

STEP 1: The data sample is Snippet $S = (w_1, w_2, \ldots, w_n)$, where, w_i for $i=1 \ldots n$ are the words in the snippet after removing stop words.

STEP 2: Suppose that there are m categories $C_1, C_2, \ldots, C_m$. Given an unknown Snippet S (i.e., with no label), the classifier will predict that S belongs to the category having highest probability.

By Bayes Theorem $P(C_i/S) = P(S/C_i) * P(C_i) / P(S)$

STEP 3: $P(S)$ is constant for all categories, therefore only $P(S/C_i) * P(C_i)$ need to be maximized.

If the category prior probabilities are not known, then it is assumed that categories are equally likely i.e., $P(C_1) = P(C_2) = \ldots = P(C_m)$. Therefore maximize only $P(S/C_i)$.

Otherwise, maximize $P(S/C_i) * P(C_i)$. This even identifies the user's interest in a particular category.

Category prior probability is estimated as

$P(C_i)$ = (Number of times user visited category C_i / Total number of times visits)

STEP 4: Now, Naïve assumption of *class conditional independence* is made.

i.e., $P(S/C_i) = \prod_{k=1}^{n} P(S_k/C_i)$

Thus $P(w_1/C_i), P(w_2/C_i), \ldots, P(w_n/C_i)$ can be estimated from training samples, where $P(S_k/C_i)$ = (Number of words in C_i having value S_k / number of words in C_i).

STEP 5: In order to classify an unknown snippet S, $P(S/C_i) * P(C_i)$ is evaluated for each category C_i. Sample snippet S is then assigned to the category C_i if and only if $P(S/C_i) * P(C_i) > P(S/C_j) * P(C_j)$ for $1 \le j \le m, j \ne i$.

In other words, it is assigned to the category C_i for which $P(S/C_i) * P(C_i)$ is maximum.

Observation: Our version of Naïve Bayesian Probabilistic model (Table 1) differs in computation of category prior probability $P(C_i)$ (Step 3). Traditional model esti-

mates category prior probability by $P(C_i)$ = (number of training samples in category C_i / total number of training samples), instead we make use of user profile for computation of $P(C_i)$. Also we classify the sites into the categories found in the *user profile* (Table 1) and not rest, since user would be most interested in those categories found in his profile. This reduces the cost of the model to a large extent because we are not classifying the sites into all the available categories. It only tries to classify into other categories when user is no more interested in these categories, which seldom happens. In that case $P(C_i)$ is not considered, i.e., only $P(S/C_i)$ is maximised. Three cases exist:

Case 1: YES, the results retrieved are as needed by the user.
- Increment the weight of that particular category, $C_{Weight} = C_{Weight} + 1$.

Case 2: NO, the results retrieved are not as per user's need.
- Classify the remaining results and display.
- Append the category of the site on which user clicks and assign weight 1.

Case 3: YES, the retrieved results are correct but user wants other results also.
- Increment the weight of that particular category, $C_{Weight} = C_{Weight} + 1$.
- Classify the remaining results and display.
- Append the category of the site on which user clicks and assign weight 1.

Table 2: User Profile.

Category (C_i)	Weight (No. of visits)	$P(C_i)$
Science > Astronomy	2	2/10 = 0.2
Science > Computer	5	5/10 = 0.5
Science > Biology	3	3/10 = 0.3

Example: Lets assume that a user has logged in and browsed 10 sites, 5 of which fall under Science > Computer, 3 of which come under Science > Biology and remaining 2 Science > Astronomy as shown in his profile (Table 2). Now if he submits a query say 'Genetics' (Ambiguous because genetics can be related to both genetic algorithms and genetics of biology), all the possible results of genetics will be retrieved initially. Now we apply Naïve Bayesian algorithm to classify these results into categories found in the profile.

Let's say there is an unknown snippet, which is to be classified. It has words w_i = {genetics, powerful} after getting pruned. Number of training samples belonging to category Science > Astronomy (SA) = 5, neither of the words w_i are found in training set of SA so not this category. Number of training samples belonging to category Science > Biology (SB) = 10. This has only one word, genetics. P(genetics/SB) * P(SB) = (1/10) * 0.3 = 0.03. Number of training samples belonging to category Science > Com-

puter (SC) = 8. This has only two words. genetics and algorithm. P(genetics/SC) * P(SC) = (1/8) * 0.5 = 0.0625. P(genetics/SC) * P(SC) > P(genetics/SB) * P(SB) therefore the snippet S belongs to SC i.e., (Science>Computer) category.

4. Performance Analysis

Experiment 1: Average rank of the Google search results remain the same irrespective of user interest, where as in our proposed system the ranking differs as the users and their interest changes. When user interest is 0 i.e., the system doesn't know anything about the user, the rank is same as the Google rank say 5. As user interest approaches 1, i.e., the system is learning about the user gradually, then its ranking improves to 3 as shown in Figure 2. Google pagerank works on the basis that if a website abc.com has been linked from a website xyz.com, abc.com must have some good content and therefore Google will count the link from xyz.com as a vote for abc.com.

Experiment 2: Precision is the percentage of retrieved documents that are in fact relevant to the query (i.e., "correct" responses). Precision of the system depends on the number of times the users uses the system and the sequence in which he proceeds. The precision is calculated as follows.

$$\text{Precision} = |\{\text{Number of Relevant documents}\} \cap \{\text{Number of Retrieved documents}\}| / |\{\text{Number of Retrieved documents}\}| \quad (1)$$

Fig. 2. the Precision of Google's search results is compared with our results with respect to varying logins. Precision of Google search doesn't vary much, where as the precision of our search result grows as the number times the user logins.

An illustrative example is given below to explain the variation in precision. Here we assume that user submits just one query per login.

Let's say the system retrieves 100 results every time on the query Pluto. For the first time 20 relevant documents (user interested in cartoon) are found and user clicks on cartoon's site and hence the precision is 0.2. For the second time when the user logs in, he gives a query Pluto, only cartoon sites are retrieved as user showed interest in cartoons last time. But now user doesn't want cartoons site, he wants planets. Therefore no relevant sites are found for planets sites and hence the precision is 0. For the third time when the user logs in, he again gives a query Pluto, the system retrieves both cartoons and planets because both categories have equal weight, and user is interested in any one of them, say user shows interest in cartoons, so cartoon site gets more weight and hence precision is 0.5. For the forth time when the user logs in and query is Pluto again, the system retrieves 75 cartoon results and 25 planet results based on weight. Users wants cartoon and therefore precision is 0.75. In the next login with the same

query Pluto the system retrieves 80 results of cartoon and 20 of planets and user wants cartoon sites and hence precision is 0.8.

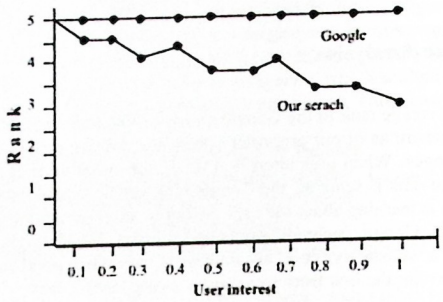


Fig. 3. Variation in ranking of Google results and Our results as the user interest is varied.

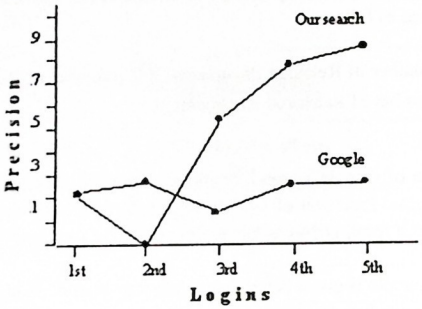


Fig. 4. : Variation in precision of Google results and our search results as the number of times user logs in.

Experiment 3: Recall is the percentage of documents that are relevant to the query and are, in fact, retrieved. It is formally defined as,

Recall = | {Number of Relevant documents} ∩ {Number of Retrieved Documents} | / | {Number of Relevant documents} |. (2)

Figure 5 shows a variation in recall of Google results and our search results as the user interest is varied. Recall of Google search results doesn't change much as it doesn't

depend on user's interests whereas recall in our case varies as user interest change and it approaches one as user interest approaches one.

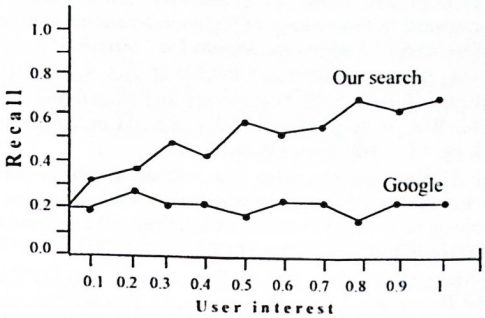


Fig. 5. Variation in recall of as the user interest is varied.

5. Conclusions

In this paper, we have proposed a new version of Naive Bayesian probabilistic model, which classifies the sites into different categories. The algorithm is tested on Google's results and Google PageRank to obtain the top 10 results and the results obtained are satisfactory in nature. Table 4 shows the Comparison of Google search results with our search results for the query Pluto. Table 5 Comparison of Google search results with our search results for the query Genetic.

Acknowledgements

This work is partially supported by AICTE, New Delhi, under AICTE Career Award for Young Teachers (AICTE File No. F.No.1-51/FD/CA/(9)/2005-06) to Mr. Srinivasa K G, Lecturer, Department of Computer Science and Engineering, M S Ramaiah Institute of Technology, MSR Nagar, Bangalore – 560054, Karnataka, India.

References

[1] Stefan Decker, Michael Erdmann, Dieter Fensel, Rudi Studer, (1998). Ontobroker: Ontology based Access to Distributed and Semi-Structured Information, In Proceedings of W3C Query Language Workshop QL'98.

[2] Satochi Oyama, Takashi Kokubo, and Toru Ishida, (2004). Domain-Specific Web Search with Keyword Spices, 1041-4347/04/ 2004 IEEE.

[3] Sandip Debnath, Prasenjit Mitra, Nirmal Pal, and C. Lee Giles (2005). Automatic Identification of Informative Section of Web Pages, 1041-4347/05 IEEE.

[4] Labrou, Finin (1999). Yahoo! as an ontology: using Yahoo! categories to describe documents. In Proceedings of Eighth international conference on Information and Knowledge Management, Kansas City, Missouri.

[5] Jude Shavlik, Susan Calcari, Tina Eliassi-Rad, Jack Solock (1999). An Instructable, Adaptive Interface for Discovering and Monitoring Information on the World Wide Web. In Proceedings of International Conference on Intelligent User Interfaces, pp. 157 - 160, Redondo Beach, CA.

[6] Junzhong Ji, Chunnian Liu, Jing Yan (2004). Bayesian Networks Structure Learning and Its Application to Personalized Recommendation in a B2C Portal, In Proceedings of the IEEE/WIC/ACM International conference on Web Intelligence (WI'04), 0-7695-2100-2/04 IEEE.

[7] Chunyan Miao, Qiang Yana, Haijing Fang, Angela Goh (2002). Fuzy Cognitive Agents for Personalized Recommendation, In Proceedings of the 3rd International Conference on Web Information Systems Engineering (WISE.02) 0-7695-1766-8/02 IEEE.

[8] Mamata Jenamani, Pratap K.J. Mojapatra, and Sujoy Ghose (2002). Online Customized Index Synthesis in Commercial Web Sites, 1094-7167/02 IEEE Intelligent Systems, pp 20-26.

[9] Gediminas, Adomavicius, Alexander, Tuzhilin (2001). Using Data Mining Methods to Build Customer Profiles, 0018-9162/01 IEEE Computer, 74-82.

[10] Alexander Maedche and Steffen Staab (2001). Ontology Learning for the Semantic Web, 1094-7167/01 IEEE Intelligent Systems.

[11] Dwi H. Widyantoro & Thomas R. Ioerger, John Yen (2003). Tracking Changes in User Interests with a Few Relevance Judgments, CIKM'03, November 3-8, 2003, New Orleans, Louisiana, USA, ACM 1-58113-723-0/03/0011.

[12] Open Directory Project http:/dmoz.org

[13] http:/www.google.com/api

[14] Yannis Labrou, Tim Finin (1999). Yahoo! As An Ontology – Using Yahoo! Categories To Describe Documents, In Proceedings of the 8th International Conference On Information Knowledge Management (CIKM), 1999, pp. 180-187.

[15] Yahoo! http://www.yahoo.com

[16] Robert Krovetz and Bruce W. Croft (1992). Lexical Ambiguity and Information Retrieval, In Proceedings of ACM Transactions on Information Systems, 10(2), April 1992, pp. 115-141.

[17] P Deepa Shenoy, K G Srinivasa, A O Thomas, Venugopal K R & L M Patnaik (2004). Mining Top-k Ranked Webpages using Simulated Annealing & Genetic Algorithm, AACC 2004, pp. 137-144.

[18] Srinivasa K G, P Deepa Shenoy, Venugopal K R & L M Patnaik (2005). A Hybrid System for Web Search Personalization using Query Refinement, Recommender Systems and User Profiles", In Proceedings of 13th International conference on Advanced Computing (ADCOM 2005), Coimbatore, Dec. 14-17, 2005.

PM10 Classification with SOM Neural Networks for Air Quality Levels Estimation

M.G. Cortina [1], J.M. Barron [1], S. Ledesma [1], D. Andina [2] and A. Vega-Corona [1]
[1] Universidad de Guanajuato
Facultad de Ingeniería Mecánica, Eléctrica y Electrónica.
Salamanca, Guanajuato. México
januchs,badamem,selo,tono@salamanca.ugto.mx
[2] Universidad Politécnica de Madrid
Departamento de Señales, Sistemas y Radiocomunicaciones, E.T.S.I.
Telecomunicación, Madrid, Spain
andina@gc.ssr.upm.es
(Paper received on September 04, 2006, accepted on September 27, 2006)

Abstract. This paper presents a new methodology to detect and classify PM10 Particles concentration according to Air Quality Levels. In this work, Meteorological Variables are analyzed to make a classification decision. The method consists of tree steps. In first step, we group using a SOM Neural Networks the Pollutant concentrations in two classes, these classes are noise data and validated data. In second step, we create a Representative Feature Vector using the information of contingency levels that we know a priori. In third step, a new SOM Neural Network is trained with the Representative Feature Vector built in second step, and the pollutant concentrations and meteorological variables (Validated Data) are self-organized in fourth classes. Finally, we obtained the Air Quality Level. Our experiments with this methodology exhibit good results in Air Quality Classification Levels. In this case a time series obtained from the Environmental Monitoring Network of the Salamanca city, Guanajuato, México is used.

1. Introduction

Air pollution is a broad term applied to any chemical, physical, or biological agent that modifies the natural characteristics of the atmosphere. The atmosphere is a complex, dynamic natural system that is essential to support life on planet earth.

Air pollution is one of the most important environmental problems, pollution is caused by both natural and man-made sources. Major man-made sources of ambient air pollution include industries, automobiles, domestic activities and power generation [1]. Air pollution has both acute and chronic effects on human health. Health effects range anywhere from minor eyes irritations and the upper respiratory system to chronic respiratory disease, heart disease, lung cancer, and death.

Nowadays, many countries make big efforts to minimize air pollution. In polluted countries like Mexico a continuous monitoring of Air Quality to measure pollutant concentrations to reduce possible negative effects in population health is necessary. A special case with great pollution is Salamanca, Guanajuato in Mexico. Salamanca city is catalogued as one of the most polluted cities in Mexico [2]. The main causes of pollution in Salamanca are due to fixed emission sources such as Chemical Industry

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 97-104

and Electricity Generation, being the most important pollutants in Air, Sulphur Dioxide (SO_2), measured in Parts Per Billion (PPB), and Particulate Matter less than 10 micrometers in diameter (PM_{10}), measured in micrometers (μm). This article focuses on PM_{10} concentration.

In 1999, an Environmental Monitoring Network (EMN) was established, this network provided time series about criteria pollutant [3] among other meteorological variables. In an effort to fight pollution on the region, in July 2005, the Environmental Contingency Program was launched, the purpose of it being to protect population health, especially that of vulnerable groups. This program contemplates the urgent and immediate reduction of SO_2 and PM_{10} emissions when measurements of these pollutants register levels above those established by Health Authorities. To accomplish it, 3 phases were established: Pre-contingency, Phase I Contingency and Phase II Contingency for SO_2, PM_{10} particles and for a combination of both [2, 4].

1.1. Particulate Matter

Particulate Matter consists of solid or liquid aerosol particles suspended in the air and has a diverse chemical composition related to its sources. Under normal ambient conditions of sampling and analysis, particulate matter exists almost exclusively in solid phase but can include liquid aerosols such as the heavier components of diesel combustion products and nitric acid.

Some particles are emitted directly into the air from a variety of sources that are either natural or related to human activity. Natural sources include bushfires, dust storms, pollens and sea spray. Those related to human activity include motor vehicle emissions, industrial processes, unpaved roads and wood heaters. Particulate Matter is commonly designated as $PM_{2.5}$ or PM_{10} they are, refereed as particles with aerodynamic diameters less than 2.5 μm and 10 μm, respectively.

The statistical correlation between high levels of inhalable particulate matter and increased mortality has been widely reported. The particles in the $PM_{2.5}$ and PM_{10} fractions can be inhaled into the lungs, causing damage to the alveolar tissues and inducing various health problems [5,6]. The adverse effects may range from the irritation of the lung tissues resulting in coughing to severe respiratory problems for individuals with asthma or heart disease. The mechanic details of how the constituents of the particulate matter induce adverse health effects are currently areas of intense scientific research. The polycyclic aromatic hydrocarbons and heavy metals present in the PM2.5 and PM10 samples have been studied extensively with regard to their roles in inducing toxicity effects [7].

1.2. Artificial Neural Network

Artificial Neural Networks (ANN) are biologically inspired network based on the organization of neurons and decision making process in the human brain [8]. In other words, it is the mathematical analogue of the human nervous system.

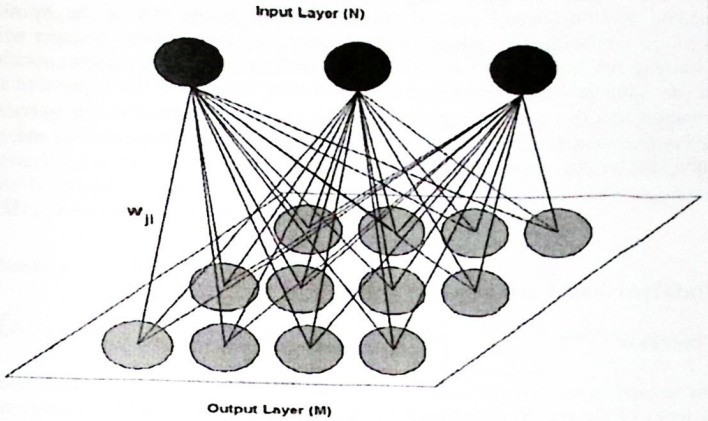


Fig. 1. SOM Neural Network structure

SOM Neural Network The Self-organizing map [9] basically provides a form of cluster analysis by producing a mapping of high-dimensional input data x, $x \in \Re^n$ onto a usually 2-dimensional output space while preserving the topological relationships between the input data items as faithfully as possible. It consists of a set of units, which are arranged in some topology where the most common choice is a two dimensional grid [10,11]. Each of the units *i* is assigned a weight vector mi of the same dimension as the input data, $m_i \in \Re^n$. In the initial setup of the model prior to training, the weight vectors are filled with random values.

During each learning step, the unit *c* with the highest activity level, the winner *c* with respect to a randomly selected input pattern *x*, is adapted in a way that it will exhibit an even higher activity level at future presentations of that specific input pattern. Commonly, the activity level of a unit is based on the Euclidean distance between the input pattern and that unit's weight vector. The unit showing the lowest Euclidean distance between its weight vector and the presented input vector is selected as winner. Hence, the selection or the winner *c* may be written as given in expression (1)

$$c : \|x - m_c\| = min_i \|x - m_i\| \tag{1}$$

Adaptation takes place at each learning iteration and is performed as a gradual reduction of the difference between the respective components of the input vector and the weight vector. The amount of adaptation is guided by a learning-rate α that is gradually decreasing in the course of time. As an extension to standard competitive learning, units in a time-varying and gradually decreasing neighborhood around the

winner are adapted, too. This strategy enables the formation of large clusters in the beginning and fine-grained input discrimination towards the end of the learning process. In combining these principles of self-organizing map training, we may write the learning rule as given in Expression 2. We make use of a discrete time notation with t denoting the current learning iteration. The other variables of this expression are α representing the time-varying learning-rate, h_{ci} representing the time-varying neighborhood-kernel, x representing the currently presented input pattern, and mi denoting the weight vector assigned to unit i.

$$m_i(t+1) = m_i(t) + \alpha(t)h_{ci}[x(t) - m_i(t)] \tag{2}$$

2. Database and Variables Definition

2.1. Database

In this research, a real and historical time series database from the EMN has been used. Data series of the months from March to May in the year 2005 have been analyzed. These time series consist of a total of 129,600 multidimensional patterns of pollutants and meteorological variables.

2.2 Variables Definition

It consists of normalized pollutants concentration values (PM_{10}) and normalized meteorological values (Wind Speed and Relative Humidity). In Table 1, variables are defined in order to build a Representative Feature Vector $\mathbf{x}_j$ and to define a pattern set $\mathbf{X}_* = \{x_1, x_2, \ldots x_j, \ldots, x_n\}$. Let $\mathbf{X}_{PM10}$ be a particles concentration set, thus their corresponding pattern is defined as $x_j = \{x_1, x_2, x_3\}$.

Table 1. Variables Definition: PM_{10} Concentration, RH; Relative Humidity, WS; Wind speed.

	$\mathbf{X}_{PM_{10}}$
Variables x_i	$\mathbf{x}_j$
x_1	PM_{10}
x_2	HR
x_3	WS

3. Proposed Methodology

Figure 2 shows the flow diagram of the methodology that was followed for the classification process. This process has three steps. Steps which will be explained in detail:

- The Cleaning Clustering Method
- The Building Representative Feature Vector Method
- The Clustering Classification Method

Cleaning Clustering Method: This method uses a SOM Neural Network to classify the patterns to be analyzed. Because time series have noise, we trained the SOM Neural Network using time series in two classes, the first class is noisy data and the second class is validated data. Validated data will be used in the Clustering Classification Method, meanwhile noisy data will be deleted.

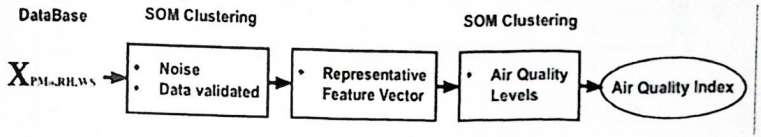


Fig. 2. Block diagram of our methodology.

Building Feature Vector Because of data nature of the variables, it is possible to know the variables limits. In this section a Representative Feature Vector (RFV) is built [12], with the centers of the involved variables, these centers are for Non Contingency, Pre-Contingency, Phase I Contingency and Phase II Contingency. This RFV was used to train a new SOM Neural Network with a [4 1] line topology, and try to locate the SOM Neural Neural node in the appropriate contingency level.

Clustering Classification Method After the SOM Neural Network has been trained with a [4 1] line topology [9,13] in order to have four clusters and therefore four prototypes according to contingency levels, we proceed to group the new patterns set in four classes according to the contingency levels (see Table 2).

Tabla 2. Contingency Concern Levels and Pattern centers vector

Contingency Level Center		
Contingency Levels	Pattern Center WS-PM_{10}	Pattern Center RH-PM_{10}
Non Contingency	75.10	75.50
Pre-Contingency	200.5.10	200.5.50
Phase I Contingency	304.5.10	304.5.50
Phase II Contingency	450.10	450.50

4 Experimental Results

Figure 3 shows the SOM Neural Network trained with the RFV shown in Table 3. Figure 4 shows a real time series for PM10 concentrations analyzed every minute.

They are classified according to the process followed by our methodology. The meteorological variables used were Wind Speed and Relative Humidity, creating 129,600 three dimensional pattern vectors x_i, for pollutant (X_{PM10}), as it is shown in Table 1. Pollutant concentrations and meteorological variables are provided by the EMN from Salamanca. The clustering method, four clusters have been performed from a new feature vector.

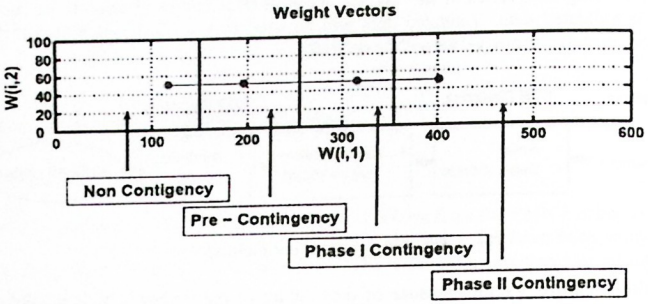


Fig. 3. SOM Neural Network structure trained using a Representative Feature Vector, with Linear [4 1] topology

Tabla 3. Contingency Concern Levels and SOM center clusters

Contingency Level Center		
Contingency Levels	Cluster Center WS-PM_{10}	Cluster Center RH-PM_{10}
Non Contingency	117.47.10	117.47.50
Pre-Contingency	196.14.10	196.14.50
Phase I Contingency	315.35.10	315.35.50
Phase II Contingency	401.53.10	401.53.50

In Table 4 the error percent obtained with the proposed methodology is shown, we can observe that the error depends on the number of data; more data, more error.

Noisy pattern is an inconsistent element in the time series and it is caused by blasts of wind. Noisy elements can cause bad estimation, so with this method a better estimate is obtained.

5. Conclusions

In this work, A time series was obtained from the EMN, this time series contain noisy data and valid data, with a SOM Neural Network this time series was clustered in two classes, the class with noisy data was deleted and the valid data was using to train a

new SOM Neural Network. This SOM classify the valid data according to the Air Contingency Levels. This methodology presented good results, because the Contingency Levels are known, allowing to create a Representative Feature Vector for each level. Thus, less patterns are required to train a SOM Neural Network. The classification error depends only on the number of data. Our method produces errors that are less than 1%.

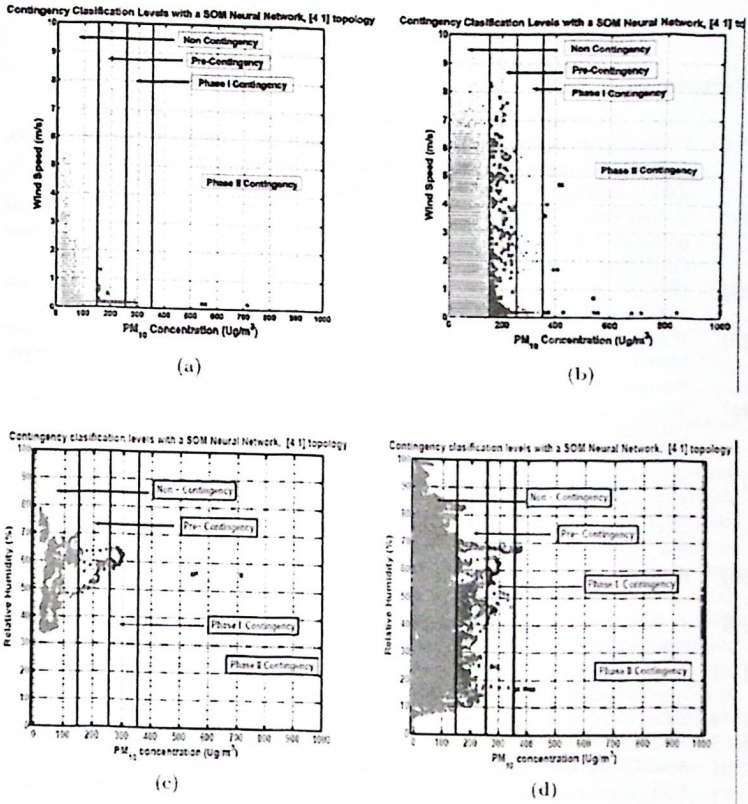


Fig. 4. (a) PM_{10} vs WS Patterns Classification for March 5, 2005. (b) PM10 vs WS Patterns Classification for March, 2005. (c) PM10 vs RH Patterns Classification for March 5, 2005. (d) PM10 vs RH Patterns Classification for March, 2005.

Tabla 4. Error percent using a SOM Neural Network trained with a Representative Feature Vector

% Error			
Data Analyzed	Number of Data	% Error WS-PM_{10}	% Error RH-PM_{10}
one day	1440	0.28	0.28
one month	32971	0.4	0.4

References

[1] Environmed Research Inc. Alpha Nutrition. Problem air pollution. http://www.nutramed.com/environment/particles.htm, 2004.

[2] SIMA. Sistema de información ambiental. http://www.sima.com.mx/, 2004.

[3] United States Environmental Protection Agency. Risk assessment for toxic air pollutants: A citizen's guide-epa 450/3-90-024. Air Risk Information Support Center (MD-13), March 1991.

[4] Instituto de Ecologia del Estado de Guanajuato. Programa de contingencias ambientales atmosféricas, 2005.

[5] Wordl Health Organization. Health Aspect of Air Pollution with Particulate Matter, Ozone and Nitrogen Dioxide. WHO Regional Office for Europe, January 2003.

[6] Secretariat of Commission for Environmental Cooperation. Continental Pollutant Pathway. Communication and Public Outreach Deppartment of CEC Secretariat, 1997.

[7] Ngee-Sing Chong, Kavitha Sivaramakrishnan, Marion Well, and Kathy Jones. Characrerization of inhalable particulate matter in ambient air by scanning electron macroscopy and energy-dispersive x-rau analysis. EJEAFChe, (3):145–164, 2002.

[8] R. Setiono and Huan Liu. Neural-network feature selector. IEEE Transactions on Neural Networks, 8(3):654–662, May 1997.

[9] T. Kohonen. The self organization map. Proceedings. IEEE, 78(9):1464–1480, September 1990.

[10] James A. Freman and David M. Skacura. Neural Networks Algorithms, Application and Programming Techniques. 1991.

[11] S.Haykin. Neural Networks. Prentice Hall, 2nd edition, 1999.

[12] Jain A.K. Pattern Recognition. John Wiley and Sons, Inc, 1988. pp.1052–1063.

[13] MatLab. Neural networks toolbox. Math Works, September 2005.

Effect of Long-delay Connections to Realize a Self-Organization of Firing Patterns that Vary Smoothly in Space

Toshio Akimitsu[1], Yoichi Okabe[2], and Akira Hirose[1]

[1] Department of Electrical Engineering The University of Tokyo, Bunkyo-ku, Tokyo 113-8656, Japan
toshio@eis.t.u-tokyo.ac.jp
[2] The University of the Air, Chiba City, Chiba 261-8586, Japan
(Paper received on August 03, 2006, accepted on September 26, 2006)

Abstract. Many experimental results suggest that more precise spike timing is significant in neural information processing. We construct a self-organization model using spatiotemporal patterns, where Spike-Timing Dependent Plasticity (STDP) tunes the conduction delays between neurons. The recurrent connections with a specific conduction delays are strengthened when the input patterns are changing at a certain period. We show that, by cooperating with short delays, long-delay connections realize topological mapping where the firing clusters are changing smoothly in space.

1 Introduction

As a well-known clustering network in brain, there are column organizations in cerebral cortex, where neurons are arranged to preserve sensory topological structure [1]. Recently experimental evidence from several different preparations suggests that both the direction and magnitude of synaptic modification arising from repeated paring of pre- and postsynaptic action potentials depend on the relative spike timing[2]. Song et al showed that an orderly topological map can arise solely through Spike-Timing Dependent Plasticity (STDP) from random initial conditions without global constraints on synaptic efficiencies, or additional forms of plasticity[3]. However, despite using the millisecond-scale model, the patterns are composed of high firing-rate Poisson inputs, and the meaning of temporal causal relationship is not clear.

On the other hand, many experiment results suggested that the more precise spike timing accomplish the key role in the brain. For example, multiunit recording studies from the frontal cortex of behaving monkey suggested that a spatiotemporal pattern of highly synchronous firing of neural populations can propagate through several tens of synaptic connections without losing high synchronicity[4]. This phenomenon is called "synfire-chain", which brings to that the neurons with long time-constant can convey the information keeping with the precise spike-time information. Diesmann et al showed that through multi-layered Feed-Forward network with Integrate & Fire neuron model, the pulse

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 105-114

packet can propagate stably in the presence of background noise if the number of neurons in a pool is large enough and yet the igniting pulse packet is synchronized strong enough [5]. This propagation is due to exact timing of the excitatory inputs. Hence, it reflects temporal coding. Additionally, several studies provide that the heterogeneous structure of the network such as the Mexican-Hat-type connectivity can convey the quantitative information [6]. Similarly, Aviel et al showed that by adding inhibitory pool, synfire-chain can be embedded in a balanced network [7]. This network can also utilize the quantitative information.

In primary visual cortex (V1), responses of neurons to a stimulus presented in their receptive fields are modulated by another stimulus concurrently presented in their surrounds. Such a contextual modulation suggests an interaction between feedback connections to V1 from other areas [8].

In this paper, we show that, by using localized synfire-chain patterns, STDP can strengthen the synaptic efficiencies having specific conduction delays and then form self-organizing map whose patterns are expressed as firing clusters. Next, we consider the case that a network is composed of synaptic connections with two types of conduction delays. We show that, by cooperating with short delays, long-delay connections cause topological mapping where the firing clusters are changing smoothly in space.

2 Model

We use a simple Integrate & Fire neuron model, and the membrane potential V is determined as

$$\tau_V \dot{V} = -(V - V_S) + J_E(t)(V - V_E) + J_I(t)(V - V_I)$$

with $V_s = V_I = -70.0\text{mV}$, $V_E = 0.0\text{mV}$, and $\tau_V = 5\text{ms}$. The synaptic inputs G_E and G_I are expressed as spatiotemporal integration of synaptic efficiencies characterized by step rise time and exponential decay

$$J(t) = \sum_j W_{ij} \sum_{t_j^k} \Theta(t - t_j^k) \exp(-(t - t_j^k)/\tau_\cdot) \quad (\cdot = E, I) \tag{1}$$

where $\Theta(t)$ is step function and the time-constant is chosen as $\tau_E = \tau_I = 5.0\text{ms}$. The synaptic strength W_{ij} is a transmission efficiency of the connection. All efficiencies from inhibitory neurons are assumed to have negative values (Inhibitory synapses), while all from excitatory ones are positive (Excitatory synapses). W_{ij}^I corresponding to the inhibitory ones are chosen as the constant values whose range is [0.18, 0.22]. On the other hand, W_{ij}^E correspond to excitatory neurons are modified via STDP whose range is [0, 0.05]. When the membrane potential V reaches a threshold value $V_{thr} = -54\text{mV}$, the neuron fires and the membrane potential is reset to $V_{res} = -60\text{mV}$. After firing, $G_{E,I}$ is kept zero during 3ms (absolute refractory period). On these conditions, about 20 coincident excitatory spikes elicit firing.

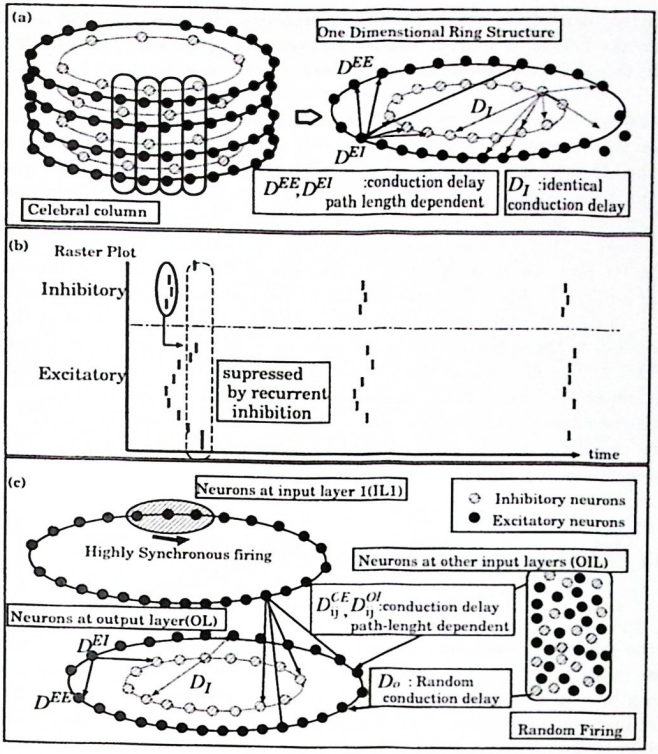


Fig. 1. (a) A network model.(b) Typical behavior of this network. (c) The whole system of this model. There are two types of inputs.

The model neural network is schematically shown in Fig. 1(a)[9]. In many brain areas, the temporal precision of spikes during stimulus-locked responses can be in the millisecond range. Reproducible temporal structure can also be found. Therefore, "delay tuning mechanism" is needed. From this viewpoint, we regard the role of STDP as a tuning of the conduction delays of the neurons. Some experiments results in cerebral cortex suggest that local inhibitory circuits contribute to improve the orientation selectivity [10]. We, therefore, determined that the inhibitory neurons receive common inputs with the excitatory ones, and consider the case that the recurrent inhibitory conduction delays D_{ij}^{IE}, D_{ij}^{II} have an identical short value $D^I = 1.0$ms. Since the excitatory and inhibitory neurons receive inputs from common layer, their firing patterns are similar to each

other. After short delay D^I, both of them receive inhibitory recurrent spikes and suppress the firings of neurons, whose postsynaptic spike latencies are large. As a result, this network detects coincident firing neurons with short latencies (Fig. 1(b)).

We assume that the input layer 1(IL1 in Fig1(c)) has 100 excitatory neurons, of which 25 neurons fire synchronously with a small fluctuation of dispersion σ, while the other 75 neurons fire randomly (10Hz Poisson spikes). Gamma-band oscillations are widely observed in brain [11]. Therefore, we determined that input patterns are firing periodically with a 25ms interval. Each synchronized sets are represented 4 times repeatedly. Hence, the patterns are shifted at 100ms interval. At first, we chose a set of 25 neurons as the spatially continuous ones forming a pattern. After the interval, the 25 neurons are shifted. This condition yields a continuously changing pattern in which each center position of the neurons represents the stimuli. This is a simple case, while general case is discussed in Section 3. The neurons at the Output layers (OL) also receive 100 excitatory and 50 inhibitory inputs firing randomly at other input layers (OILs).

We determine the conduction delay D_{ij}^{OE}, from input neuron i at IL1 to excitatory neuron j at OL, proportional to distance in such a manner that the periodical boundary condition is satisfied. That is

$$D_{ij}^{OE} \propto |i-j|_{\bmod N} \tag{2}$$

where i and j are neuron indices, and we define

$$|i-j|_{\bmod N} \equiv \min(|i-j|, N-|i-j|) \tag{3}$$

Since the number of inhibitory output neurons is half of the input-neurons number, the delay D_{ij}^{OI} from input neuron i at IL1 to the inhibitory neuron j at OL is determined to be proportion to $|i-2j|_{\bmod N}$, which should also satisfy periodical boundary condition. The maximum conduction delay is 3ms, while the minimum is 0ms. The probability that an input neuron is connected to output neuron is 0.8, and the initial values of the synaptic strength are chosen about the

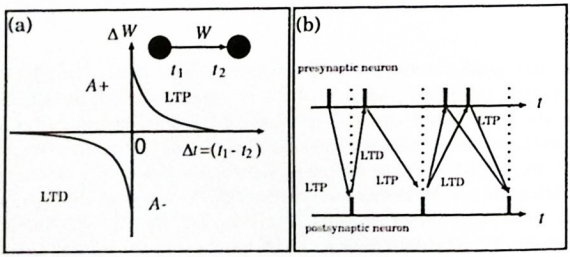


Fig. 2. (a) Window function of STDP.(b) Time Diagram showing that only spike pairs connected by arrows contribute to plasticity (near-neighbor interaction).

half of the maximum. STDP was implemented only for the excitatory synapses of the output layer's neurons,

$$\Delta W = \begin{cases} A_+ \exp(-\Delta t/\tau_+) & \Delta t > 0 \\ A_- \exp(-\Delta t/\tau_-) & \text{otherwise} \end{cases} \quad (4)$$

where A_+ and A_- are the sizes of the synaptic modification by a single STDP event. We chose $A_+ = 0.02$, $A_- = 0.025$, and $\tau_+ = \tau_- = 20$ms. LTD is implemented only after the latest firing and LTP is implemented after the last firing (near-neighbor interaction). When one firing pattern is presented, the input spikes elicit a postsynaptic response, triggering the STDP rule. Synapses carrying input spikes just preceding the postsynaptic ones are potentiated, later ones are depressed. This modification causes a decrease of the postsynaptic spike latency. Hence, at the next time, when this input pattern is presented, firing threshold will be reached sooner. Consequently, some synapses it had previously potentiated are depressed and different synapses are reinforced which carry even earlier spikes than the preceding time. By iteration, the postsynaptic spike latency will tend to stabilize at a minimal value while the first synapses become fully potentiated and later ones fully depressed[12].

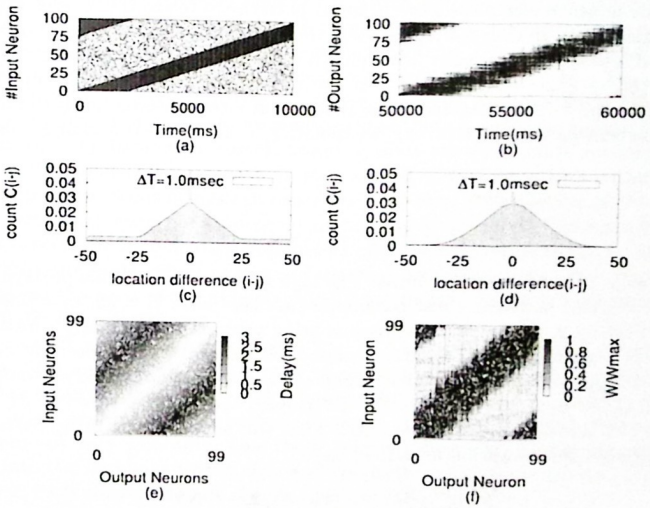


Fig. 3. (a) Spike raster showing the activity of the neurons at Input Layer 1. (b) Raster plot of excitatory neurons at Output Layer. (c)(d) Coincident clustered histgram of (a),(b). (e) Conduction delays map. (f) Weight distribution.

In this network, inhibitory neurons receive similar inputs to excitatory neurons and excitatory synapses of inhibitory ones are modified. Therefore, the changes of the postsynaptic spike latencies of inhibitory neurons are almost in keeping with excitatory ones. Therefore, this network can work as a coincidence detector, even if the synaptic efficiency has changed during learning.

To investigate a degree of spatiotemporal clustering, we calculated " coincident clustering histogram". If a difference of firing time of two neurons is within a time bin ΔT, we considered that these two neurons are coincident, and computed spatial difference histogram $\sum_k c_k(i-j)$,

$$c_k(i-j) = \begin{cases} 1 & |t_i^k - t_j^k| < \Delta T \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where t_i^k denotes the firing time of neuron i for input pattern k. The simulation result is shown in Fig. 3. The synapses only with shortest conduction delays survive and others were pruned. It also reduces noise firing. As a result, STDP refines the patterns.

3 Topological Map

Next, we consider the more general input patterns to relate this model with the topological map. In the topological map, the input patterns near to each other are mapped into spatially similar firing clusters. Here, we denote the firing patterns, $\xi^k = \{\xi_1^k, \cdots \xi_j^k, \cdots, \xi_N^k\}$ where j is the neuron index at IL1 (the maximum $N = 100$) and k is the pattern index switched at a certain interval. If the neuron j fires synchronously with others, we denote $\xi_j = 1$. We assume that ξ^k satisfies

$$\sum_j \xi_j^k = R \qquad (\forall k = 1, 2, \cdots, N) \tag{6}$$

$$\xi^1 \cdot \xi^2 = \xi^2 \cdot \xi^3 = \cdots = \xi^N \cdot \xi^1 = Rm \tag{7}$$

where m is the overlap of patterns. The high m means continuous pattern shift. In the previous sections, these parameters are kept with $R = 25$, $m = 0.96$.

In Section 2, the set of synchronous firing neurons was spatially continuous. Here, as a substitute for random conduction delays, we consider the case of shuffled input patterns. We choose a pair of neurons randomly, and exchange the indices of the neurons M times. We determined the shuffling times as $M = 400$ except the result of Fig. 4(a), M=100. After this shuffling, the firing pattern η^k also satisfies the condition expressed as

$$\sum_j \eta_j^k = R \qquad (\forall k = 1, 2, \cdots, N) \tag{8}$$

$$\eta^1 \cdot \eta^2 = \eta^2 \cdot \eta^3 = \cdots = \eta^N \cdot \eta^1 = Rm \tag{9}$$

We determined the dispersion of the synchronized pattern as $\sigma = 3.0$ms. These patterns with large dispersion are considered to correspond to the Poisson input

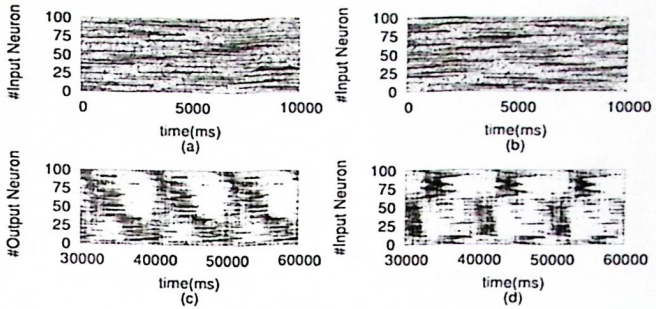


Fig. 4. Topological Mapping. Two typical output patterns are shown. (a),(b) Raster plot of the neurons at IL1. (c),(d) Raster plot of the excitatory neurons at OL.

patterns[3]. The recurrent synapse is needed to form a smoothed map. We determine the conduction delay D_{ij}^{EE} from excitatory neuron i to excitatory neuron j in proportion to $|i-j|_{\text{mod } N}$ in the range of [0.5, 1.5]. We also determined the delay D_{ij}^{EI} from excitatory neuron i to inhibitory neuron j in proportion to $|i-2j|_{\text{mod } N}$ in the same range. Hence, some of the neighboring connections are shorter than the identical inhibitory conduction delay $D^I = 1.0$ms. The initial synaptic efficiencies are also determined in proportion to $\frac{N}{2} - |i-j|_{\text{mod } N}$ in the range of $[0.25W_{\max}, 0.75W_{\max}]$, where $W_{\max}$ is the maximum efficiency ($W_{\max} = 0.05$). In this paper, we assumed that there is only one neuron in each column ($m = 1$). In cerebral cortex, however, there are much more neurons than those in this model. It is reasonable that not all the neurons in a column fire for each pattern. Some firings can cause the long-term potentiation of the other connections that were not considered in this network model. Taking this into consideration, the modification parameters of recurrent connections $B_{\pm}$ are not necessarily the same values as those of external connections $A_{\pm}$. We, therefore, determined the sizes of the synaptic modifications for recurrent connections as $B_+ = 0.02$, $B_- = 0.015$.

The result is shown in Fig. 4 and Fig. 5. Two types of results are shown. In Figs. 5(c) and (d), the gentle peak shows a synchronicity despite the asynchronous input. Figures 5 (e) and (f) show the change of the center position of the firing neurons in each pattern. Figure 5 (f) shows that the center position does not vary smoothly. This result suggests that the external stimulus is coded into the highly-synchronously and spatially-continuously clustered firing. However, such clusters do not always vary smoothly in space.

4 Interaction to different area

In this section, we investigate effects of interactions of two neurons distant from each other. We considered two types of conduction delays. For local connection,

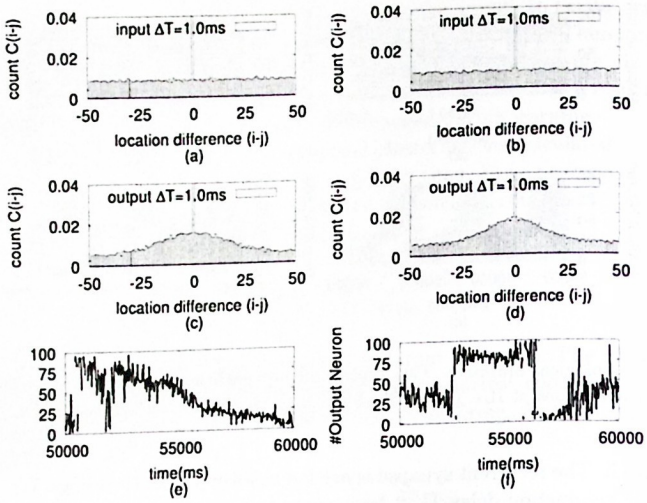


Fig. 5. (a,b) Coincident clustering histogram for input neurons. (c,d) Coincident clustering histogram for output neurons. (e,f) The center position of neurons firing in each pattern.

the synaptic delays are chosen in proportion to the location indices. With a probability p, the synapse is determined to connect to neurons to a distant area. We chose the conduction delays,

$$D_{ij}^{EE} = \begin{cases} N(1,15) & \text{Probability } p \\ |i-j|_{\bmod N} + 0.5 & \text{otherwise} \end{cases} \tag{10}$$

where N means a Gaussian and the conduction delays from distant area are determined to have a value in the range of [1, 30]. The center position of the neurons firing for the certain patterns is considered to code a kind of information such as location. We calculate a regression line of the center positions of the neurons for all patterns. Since the patterns are changing in a certain period, a trend of the recurrence is assumed to have a constant value. Therefore, we calculate a $y-$intercept value to give a minimum value of mean squared error as an index showing a smooth changing of the clusters.

$$\begin{aligned} E_+(b)^2 &\equiv \min_b \sum_k (y_i - 0.01(x_k - b))^2 \\ E_-(b)^2 &\equiv \min_b \sum_k (y_k + 0.01(x_k - b))^2 \\ E(b) &\equiv \min_b (E_+(b), E_-(b)) \end{aligned} \tag{11}$$

Figure 6 shows the result. With a small probability ($p \sim 0.1$), the clusters are

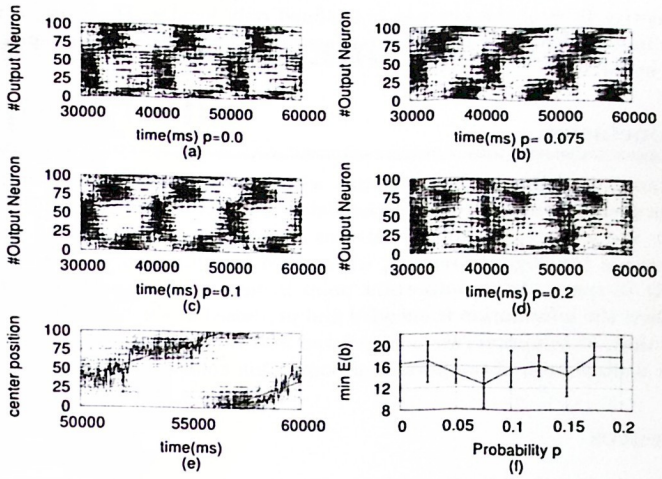


Fig. 6. (a),(b),(c),(d) Examples of raster plot of output neurons for various initial conditions. (e) A regression line for center position of the neurons. (f) Average Error for different probability values p.

changing smoothly.

5 Discussion

The neurons in a primary visual area show different behavior depending on the stimuli concurrently presented in their receptive-field surrounds. Such modulations are considered to be caused by interactions from different area. We investigate the effects of long-delay connections when the input patterns are changing in a certain period. Recurrent connections are needed to form the firing clusters. However, such clusters are not always varying smoothly in space. The long-delay connections help to interpolate clusters that might be changing discontinuously. When the neurons receive discontinuously changing patterns, the firing clusters can vary smoothly in space. Through STDP, synaptic efficiencies are modified depending only on the pre- and post-spike timing of each firing. Respective changing in efficiency can not utilize information of the whole patterns. A highly adaptable initial condition is needed to organize itself suitably to environment. We showed that an appropriate existence of long-delay connections interpolates firing clusters, which leads to the organization of firing clusters varying smoothly in space.

In pattern recognition, the network learns patterns by using only small set of samples. It is important ability for a network to retrieve patterns similar to learned ones without being affected by small perturbations. On the other hand,

the receptive field of the neurons is confined only locally. Therefore, a mechanism to modify randomly changing patterns into smoothly changing ones is also needed in the brain.

6 Conclusion

We demonstrated that self-organization with locally synchronized patterns can be performed by tuning the conduction delays between neurons. Through STDP, spatially and temporally random patterns are bunching. The long-delay connection causes a topological mapping where such bunching clusters are changing smoothly in space. As an important point in neural research, there is a question of how the information is encoded and processed in the brain. These results suggest that, as temporal rate is translated into the firing frequency, the neural network utilizes both of temporal and population codes.

References

1. LeVay, S., Wiesel, T.N., Hubel, D.H.: The Devealeopment of Ocular Dominance columns in normal and visually deprived monkeys. Journal of Comparative Neurology **191** (1980) 1–51
2. Bi, G., Poo, M.: Activity-Induced synaptic modifications in hippocampal culture: dependence on spike timing synaptic strength and cell type. Journal of Neuroscience **18** (1998) 10462–10472
3. Song, S., Abott, L.F.: Cortical Development and Remapping through Spike-Timing-Dependent Plasticity. Neuron **32** (2001) 339–350
4. Abeles, M., Bergman, H., Margalit, E., Vaadia, E.: Spatiotemporal firing patterns in the frontal cortex of behaving monkeys. J Neurophysiol **70** (1993) 1629–38
5. Diesmann, M., Gewaltig, M.O., Aertsen, A.: Stable Propagation of synchronous spiking in cortical neural network. Nature **402** (1999) 529–533
6. Hamaguchi, K., Aihara, K.: Quantitative information transfer through layers of spiking neurons connected by Mexican-Hat-type connectivity. Neurocomputing **58-60** (2004) 85–90
7. Aviel, Y., Horn, D., Abeles, M.: Synfire waves in small blanced networks. Neurocomputing **58-60** (2004) 123–127
8. Murphy, P.M., Duckett, S.G., Sillito, A.M.: Feedback connection to the lateral geniculate nucleus and cortical responce properties. Science **286** (1999) 1552–1554
9. Akimitsu, T., Hirose, A., Y.Okabe: Self-organization thorough Spike-Timing Dependnet Plasticity using localized Synfire-Chain patterns. ICONP (2006) to be presented
10. Wang, Y., Fujita, I., Murayama, Y.: Neuronal mechanisms of selectivity for object features revealed by blocking inhibition in inferotemporal cortex. Nature Neuroscience **3** (2000) 807–813
11. Neurenschwander, S., Singer, W.: Long-range synchronization of oscillatory light responses in the cat retina and lateral geniculate nucleus. Nature **379** (1996) 728–733
12. Guyonneau, R., VanRullen, R., Thorpe, S.J.: Neurons Tune to the Earliest Spikes Through STDP. Neural Computation **17** (2005) 859–879

Neural Control and Robotic Applications

Local Optimal Control Based on Neural Network Observer States

N. Aguilar, A. Cabrera and I. Chairez
Departamento de Bioelectrónica, Unidad Profesional Interdisciplinaria de Biotecnología, IPN.
Av. Acueducto de Guadalupe s/n CP. 07250, Col. Barrio la Laguna Ticoman, Mexico D. F.
e-mail: crubskay@hotmail.com
(Paper received on July 20, 2006, accepted on September 25, 2006)

Abstract. Immunotherapy refers to the use of natural and synthetic substances to stimulate the immune response. This paper provides a description for an adaptive locally optimal control design for immunotherapy cancer treatment mathematical model, where all state vector is considered to be not on-line available. The control strategy is suggested in two stages: the first one deals with the state estimation process using differential neural networks technique. The second part introduces the construction for a locally optimal control function based on the DNN mathematical representation. This technique was successfully applied in the tracking process for immunotherapy dosage control.

1. Introduction

1.1. Neural Networks with Differential Representation

The increasing demand of technology requires different approaches to solve control and identification problems of nonlinear systems. The neural networks (NN) promise better solutions in some problems. The control applications of NN are motivated for the necessity to treat more complex systems, to find new adaptive control methods and to regulate systems affected by external uncertainties. Roughly speaking, the NN can be classified as follows: *static* one, using mainly the back-propagation technique [1], and *dynamic (or recurrent) neural networks* which applies differential learning laws or dynamic (recurrent) ones [2]. Nowadays the neural networks employment in several science fields, as pattern recognition, image processing, industrial process, biological systems and automatic control engineering, has increase the interest to develop new approaches that could improve the results obtained until today. One of these new kinds of dynamic neural networks is the differential neural networks (DNN) which has been applied successfully in many identification, estimation and control techniques, such as non-parametric estimation for diabetes mellitus illness [3], fermentative process [4], [5] and Human Immunodeficiency Virus dynamics as well [4]. This approach permits to avoid many problems related to global extreme search converting the learning (training) process to an adequate state feedback design. The DNN-approach provides an effective instrument to attack a wide spectrum of problems such as identification, state estimation, trajectories tracking, if the mathematical model of a considered process is incomplete or partially known, [1]. Considering all these previous papers and the well studied identification and control theory developed in [7], a novel cancer

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 117-130

treatment control technique is suggested in order to obtain a slightly improvement in the illness evolution.

1.2. A brief review on robust output feedback control

Research on adaptive output feedback control of uncertain non-linear dynamic systems is of paramount importance today, particularly considering the growing interest in the use of unconventional control devices such as chemical sensors, piezoelectric actuators, etc. In a large number of practical problems there are important disturbances, uncertainties or several parameter variations. In these situations, a solution of the control problem is given by the, so-called, unconventional adaptive control supplied by a mechanism to adjust the controller's parameters. In the case of only one single output measurement, the existing works show semi-global results for a class of systems whose non-linearities depend on the unmeasured variable, [8] and [9]. If the non-linearity depends on output measured variables, the published papers suggest a mechanism which can achieve global results only for a class of parametric output-feedback systems. By the adaptive observer backstepping technique, a self-adjusted output feedback controller can be designed for a class of parametric output-feedback form [10] guarantying asymptotic tracking of the reference signal while keeping bounded all signals (states and control). The observer based-control is a way to solve the output feedback control problem that imposes a restriction in the possibility to use all the states directly for feedback control design. Still for linear systems, this allows to discompose the problem in two sub-problems that can be solved separately: a) by designing a state observer and b) by designing a controller based on the on-line state estimates. This approach is known as the *Separation Principle*. However, in non-linear systems, this principle is not well justified [11].

In the context of the non-linear systems, adaptive controllers that ensure good tracking performance (and disturbance rejection) of signals that corresponds to bounded solutions of known non-linear time variant systems are not clearly established. The suggested solutions for this problem are based on the internal model principle wherein the "dynamics" of the reference signal (and disturbances) are incorporated in the plant dynamics via an adequate chosen pre-filter or pre-compensator. For this study, such pre-filter going to be design as a "internal" differential neural network whose states ($\tilde{x}_t$) will be used by the controller structure to reproduce the non-linear system dynamics (represented by the supplied differential neural network observer) and then to "adjust" the control function in order to follow the reference trajectories. Using the following definitions: x_t is the current state of an uncertain system, $\hat{x}_t^{\circ}$ is the current state estimation generated by DNN observer (DNNO) which is shown to be close to x_t (that is, $x_t \sim \hat{x}_t$ by means of a special Lyapunov-like function), $\tilde{x}_t$ is the internal model state of the neural controller which is closed to $\hat{x}_t^{\circ}$ (that is, $\hat{x}_t^{\circ} \sim \tilde{x}_t$) and x_t^* is the state of a reference model. In this paper, the following fact going to be demonstrated: $x_t \sim \hat{x}_t^{\circ} \sim \tilde{x}_t \sim x_t^*$. This means that the application of a control, which

solves the problem $\left(\widetilde{x}_t \sim x_t^*\right)$, for an initial uncertain system with states x_t at the same time guarantees that $x_t \sim x_t^*$.

2. System description and basic assumptions

Let's consider the class of perturbed non-linear systems with incomplete information (not all the state vector is assumed to be available) described by the set of ordinary differential equations

$$\dot{x}_t = f\left(x_t, u_t^*, t\right) + \xi_{1,t}, \quad y_t = Cx_t + \xi_{2,t} \tag{1}$$

where $x_t \in \Re^n$ is the system state vector, $y_t \in \Re^p$ is the system output vector, $u_t^* \in \Re^m$ is the control action belonging to a specific admissible set U^{adm} that will be define below, $C \in \Re^{p \times n}$ is an a priory known output matrix. The uncertain vectors $\xi_{1,t}$ and $\xi_{2,t}$ represent the state and output deterministic external bounded (unmeasurable) disturbances $\left(\left\|\xi_{j,t}\right\|^2_{\Lambda_{\xi_j}} \le \Upsilon_j, \Lambda_{\xi_j} > 0,\ j = 1,2\right)$. Hereafter it is supposed the non-linear function in (1) satisfies the Lipschitz property (uniform on t and for all possible systems belonging to the non-linear systems class given above):

$$\begin{gathered} \left\|f\left(x,u^*,t\right) - f\left(z,v^*,t\right)\right\| \le L_1\left\|x - z\right\| + L_2\left\|u^* - v^*\right\| \\ \left\|f\left(0,0,t\right)\right\|^2 \le C_1;\ \ 0 \le L_1, L_2 < \infty;\ \ x, y \in \Re^n;\ u^*, v^* \in \Re^m \end{gathered} \tag{2}$$

The last assumption automatically implies the ODE solution existence and uniqueness and, obviously, the following property $f\left(x_t, u_t^*, t\right)^2 \le C_1 + C_2\left\|x_t\right\|^2 + C_3\left\|u_t^*\right\|^2$ that is valid for any x, u and t. Besides the unperturbed $\left(\xi_{1,t} = \xi_{2,t} = 0\right)$ non-linear system (1) is assumed to be stable, that is, there exists a Lyapunov function $\overline{V}\left(x_t\right)$ fulfilling the following inequality $\left\|\frac{d}{dx_t}\overline{V}\left(x_t\right)\right\| \le -\lambda$ and using (2) then the next expression is valid $\frac{d}{dt}\overline{V}\left(x_t\right) \le -\lambda\left[\overline{\upsilon} + \overline{C}_2\left\|x_t\right\|^2\right]$. Notice that (1) "always" could be rearranged as

$$\begin{aligned} \dot{x}_t &= f_0\left(x_t, u_t^*, t \mid \Theta(t)\right) + \widetilde{f}_{1,t} + \xi_{1,t} \\ \widetilde{f}_{1,t} &:= f\left(x_t, u_t^*, t\right) - f_0\left(x, u^*, t \mid \Theta(t)\right) \end{aligned} \tag{3}$$

where $f_0\left(x,u^*,t\mid\Theta(t)\right)$ will be referred like "*nominal dynamics*" that can be selected according to designer desires; and $\widetilde{f}_t$ is a vector defined like the "*no modelled dynamics*". Here, the set of parameters $\Theta(t)$ are subjected to adjustment (this is the so-called training process) in order to obtain the best possible accuracy in the nominal representation (a DNN representation, for example). According to DNN approach [1], the nominal dynamics is defined by

$$\begin{gathered} f_0\left(x,u^*,t\mid\Theta\right)=A^*x+W_{1,t}^*\sigma(x)+W_{2,t}^*\varphi(x)u^*,\ \Theta:=A^*,W_{1,t}^*,W_{2,t}^*\in\Re^{n\times n} \\ W_{1,t}^*\in\Re^{n\times l},\sigma(\cdot)\in\Re^{l\times 1},\ W_{2,t}^*\in\Re^{n\times s},\varphi(\cdot)\in\Re^{s\times m} \end{gathered} \tag{4}$$

The admissible control set for u_t^* (at least for the estimation process, without any feedback control design) is supposed to be bounded:

$$U^{adm}:=\left\{u^*:\left\|u^*\right\|_{\Lambda_u}^2\le v_0^2\right\}\ \Lambda_u=\Lambda_u^T>0 \tag{5}$$

The activation vector-functions $\sigma(\cdot):=\left[\sigma_j(\cdot)\right]_{j=\overline{1,l}}^T$, $\varphi(\cdot):=\left[\varphi_{jk}(\cdot)\right]_{j=\overline{1,n};k=\overline{1,s}}^T$, are usually composed with smooth monotonically growing functions. As usual, the selected activation functions are sigmoid ones:

$$\begin{aligned} \sigma_i^{-1}(x) &:= a_{\sigma i}^{-1}\left(1+b_{\sigma i}\exp\left(-\sum_{j=1}^{n}c_{\sigma i}x_i\right)\right) \\ \varphi_j^{-1}(x) &:= a_{\varphi j}^{-1}\left(1+b_{\varphi j}\exp\left(-\sum_{j=1}^{n}c_{\varphi j}x_j\right)\right) \end{aligned} \tag{6}$$

It is easy to proof they satisfy the sector conditions $\left\|\sigma(x)-\sigma(x')\right\|^2\le l_\sigma\left\|x-x'\right\|^2$ and $\left(\varphi(x)-\varphi(x')\right)u^2\le l_\varphi\left\|x-x'\right\|^2v_o^2$. In view of (2) the following upper bound for the no modelled dynamics $\widetilde{f}_{1,t}$ takes place:

$$\left\|\widetilde{f}_{1,t}\right\|_{\Lambda_f}^2\le\widetilde{f}_{1,0}+\widetilde{f}_{1,1}\left\|x_t\right\|_{\Lambda_{\widetilde{f}}}^2,\ \Lambda_f=\Lambda_f^T>0,\Lambda_{\widetilde{f}}=\Lambda_{\widetilde{f}}^T>0 \tag{7}$$

3. DNN training algorithm

The training process for the DNN algorithm is associated with the off-line best possible selection (before the begging of the state estimation) of the nominal parameters $\Theta:=\left[A^*,W_{1,t}^*,W_{2,t}^*\right]$ using experimental data $\left(x_{t_k},u_{t_k}\right)$. Notice that the

corresponding rate-vectors $\dot{x}_t$ are not available. In view of the nominal dynamics selection, the following procedure could be applied in order to define the parameter which going to be used in the non-linear output feedback controller design.

DNN trainer. In order to define the training process, all the vector state and the control action (x_{t_k}, u_{t_k}) could be measurable in specific times t_k (by direct experimentation), but the exact mathematical structure about the vector field $f(x_t, u_t^*, t)$ is partially unknown (the order of the system and its stability properties are *a* priory assumed). A special nonparametric identifier (based on DNN properties) will be applied to derive an approximation of this non-linear model. The mathematical description of this neural trainer is given by:

$$\frac{d}{dt}\hat{x}_t = A\hat{x}_t + W_{1,t}\sigma(\hat{x}) + W_{2,t}\varphi(\hat{x})u_t^* + K_1\Delta_t + K_2\frac{\Delta_t}{\|\Delta_t\|}. \tag{8}$$

Here $\Delta_t = \hat{x}_t - \bar{x}_t \in \Re^n$ is the identification error between the system states to be identified and the corresponding DNN variables, $\hat{x}_t \in \Re^n$ is the state of the neural network, $\bar{x}_t$ are reconstructed versions of x_{t_k} and u_{t_k} respectively using classical interpolation algorithms [12], $A \in \Re^{n\times n}$ is a stable matrix (maybe diagonal or upper triangular) which elements going to be selected below, $W_{1,t} \in \Re^{n\times k}$ is the weight matrix for non-linear "state feedback" and $W_{2,t} \in \Re^{n\times r}$ is the "input" weight matrix; both matrices will be adjusted using a special learning law described by matrix differential equations. The vector field $\sigma(x_t): \Re^n \to \Re^k$ and the matrix function $\varphi(\cdot)$ have the same meaning and structure that (6). In view of the neural network structure (8), it can be classified as a Hopfield-type [1]. K_1 and K_2 are constants matrices which have the same function like correction terms in observers structure [13]. These two additional terms were added following the results derived in [11] where a significant reduction in the upper bound for the convergence process was obtained. Taking into account that the elements of $\sigma(\cdot)$ and $\varphi(\cdot)$ are chosen as sigmoid functions, the following assumption are easily fulfilled.

As.1: The functions $\sigma(\cdot)$ *and* $\varphi(\cdot)$ *satisfy the Lipschitz condition:*

$$\begin{gathered}\tilde{\sigma}_t^T\Lambda_\sigma\sigma_t \le \Delta_t^T D_\sigma \Delta_t, \quad \sigma_t^T Z_\sigma \sigma_t \le x_t^T C_\sigma x_t \\ (u_t^*)^T\tilde{\varphi}_t^T\Lambda_\varphi\tilde{\varphi}_t u_t^* \le v_0^2\Delta_t^T D_\varphi\Delta_t, \quad (u_t^*)^T\tilde{\varphi}_t^T Z_\varphi\tilde{\varphi}_t u_t^* \le v_0^2 x_t^T C_\varphi x_t \\ \tilde{\sigma}_t := \sigma(\hat{x}_t) - \sigma(x_t), \quad \tilde{\phi}_t := \phi(\hat{x}_t) - \phi(x_t)\end{gathered} \tag{9}$$

and $\Lambda_\sigma, \Lambda_\phi, D_\sigma, D_\phi, Z_\sigma, Z_\phi, C_\sigma, C_\phi$ are know positive definite matrices. The

identifier design requires the next important assumption:

As.2. There exist a strictly positive defined matrix Q_0 *such that the Riccati equation*

$$A^T P + PA + PRP + Q = 0 \tag{10}$$

has a positive solution $P = P^T > 0$, where $R = \Lambda_f^{-1} + W_{1,t}^* \Lambda_\sigma^{-1} \left(W_{1,t}^*\right)^T + W_{2,t}^* \Lambda_\varphi^{-1} \left(W_{2,t}^*\right)^? + K_1 \Lambda_1^{-1} K_1^? + K_2 \Lambda_2^{-1} K_2^? + \Lambda_{\xi 1}^{-1}$ and $Q = D_\sigma + v_0 D_\varphi + \Lambda_1 + Q_0$. If the algebraic Riccati equation has positive definite solution, the weights $W_{j,t}$, $j = 1,2$ are adjusted by

$$\begin{aligned} &\frac{d}{dt}\tilde{W}_{j,t} = \frac{d}{dt} W_{j,t} = -K_j P \Delta_t \Omega_j^? \\ &\Omega_1 := \sigma(\hat{x}_t),\ \Omega_2 := \varphi(\hat{x}_t) u_t^*,\ K_j > 0,\ \tilde{W}_{j,t} = W_{j,t} - W_{j,t}^* \end{aligned} \tag{11}$$

The training process could be abstracted in the following:

Theorem 1. Let 's consider the partially unknown non-linear system (1) and a model matching neural network (8) whose weights are adjusted by the matrix differential equations (11), and also the assumptions *As.1* and *As.2* hold, then the upper bound (in average sense) for the identification (training) error is described by

$$\begin{aligned} &\overline{\lim_{T\to\infty}} \frac{1}{T} \int_{t=0}^{T} \left\| \Delta_t \right\|_P dt \le \frac{\rho_T}{\alpha_{Q,T}} \\ &\alpha_{Q,T} = \lambda_{\min}\left(P^{-1/2} Q_0 P^{-1/2}\right) > 0,\ \rho_T := \tilde{f}_{1,0} + n\left\|\Lambda_2\right\| + \Upsilon_1 \end{aligned} \tag{12}$$

The identification error sign term in (8) is introduced to apply the robustness settings on sliding mode approach, especially those that can help to solve the parameter uncertainties in the problem description. This important behaviour could be useful in this new kind of identifier because in DNN the most difficult problem is to select the matrix A. That is why the simultaneous application of DNN and sliding mode technique seems to be a promising solution to solve the problem to be tackled in this report.

4. Output feedback DNN controller

Once the DNN has been trained the automatic feedback controller for uncertain nonlinear system, affected by external perturbations could be designed in two stages, the first one deals with the state estimator develop using a special class of DNN, considering the control action as a function depending on the state estimated. The second section is related with the adaptive control suggestion; in this work three different methods are applied to solve this problem.

4.1. DNN Non-Parametric State Estimator

The observer design is going to be maddened using the ideas developed on [11]. So, let define DNN observer as follows

$$\frac{d}{dt}\hat{x}_t^\circ = A\hat{x}_t^\circ + W_{1,t}^\circ\sigma\left(\hat{x}_t^\circ\right) + W_{2,t}^\circ\varphi\left(\hat{x}_t^\circ\right)u_t + K_1^\circ\left(y_t - C\hat{x}_t^\circ\right) + K_2^\circ\frac{y_t - C\hat{x}_t^\circ}{\left\|y_t - C\hat{x}_t^\circ\right\|} \tag{13}$$

$$\hat{y}_t^\circ = C\hat{x}_t^\circ$$

Here $\hat{x}_t^\circ \in \Re^n$ is the state of the neural network observer, $W_{1,t}^\circ \in \Re^{m\times k}$ and $W_{2,t}^\circ \in \Re^{m\times r}$ are the corresponding weight matrices for this observer adjusted by the special *updating* (learning) law for the estimation process

$$\begin{gathered}\frac{d}{dt}\widetilde{W}_{j,t}^\circ = -K_j\left(P_2\hat{x}_t^\circ + P_1N_\delta\left(C^Te_t + \Pi_jN_\delta P_1\overline{W}_{j,t}\chi_j\right)\right)\chi_j^T \\ \chi_1 := \sigma(\hat{x}_t), \chi_2 := \varphi(\hat{x}_t)u_t, k_{wj} > 0 \\ \Pi_j := C^T\Lambda_{\xi_{2,t}}C + \Lambda_j^{-1}, \Lambda_j > 0;\ N_\delta := \left(CC^T + \delta I_{n\times n}\right)^{-1}, j = 1,2\end{gathered} \tag{14}$$

$P_1 = P_1^T > 0$ and $P_2 = P_2^T > 0$ are positive definite solutions (if they exist) of the following Riccati equations

$$P_h\widetilde{A}_h + \left(\widetilde{A}_h\right)^T P_h + P_hR_hP_h + Q_h = 0,\ \ h = 1,2 \tag{15}$$

whose parameters are defined by

$$\begin{gathered}\widetilde{A}_1 := \left(A - K_1^\circ C\right),\ \ \widetilde{A}_2 := A \\ Q_1 := \delta^2\Lambda_1 + \delta^2\Lambda_2 + \delta k\upsilon I_{n\times n} + D_\sigma + 2\Xi\nabla_2\Lambda_u^1 + \\ \Xi_1\nabla_2\Lambda_u^1 + 2\overline{f}_{1,1}\Lambda_f + \Lambda_3 + Q_0 \\ R_1 := W_{1,t}^\circ\Lambda_\sigma^{-1}\left(W_{1,t}^\circ\right)^T + \left(W_{2,t}^\circ\right)\Lambda_\varphi^{-1}\left(W_{2,t}^\circ\right)^T + K_1^\circ\Lambda_{\xi_{2,t}}^{-1}\left(K_1^\circ\right)^T + \Lambda_{f_{1,t}} + \Lambda_{\xi_{1,t}} \\ Q_2 := Z_\sigma + 2\Xi\nabla_2\Lambda_u^1 + 2\overline{f}_{1,1}\Lambda_f + \Xi\nabla_3\Lambda_u^2 + \nabla_3\Xi_1\Lambda_u^2 + \Xi_1\nabla_2\Lambda_u^1 \\ R_2 := K_1^\circ C\Lambda_3^{-1}C^T\left(K_1^\circ\right)^T + K_1^\circ\Lambda_{\xi_{2,t}}^{-1}\left(K_1^\circ\right)^T + K_2^\circ\Lambda_4\left[K_2^\circ\right]^T \\ + W_{1,t}^\circ Z_\sigma^{-1}\left[W_{1,t}^\circ\right]^T + W_{2,t}^\circ Z_\varphi^{-1}\left[W_{2,t}^\circ\right]^T\end{gathered} \tag{16}$$

where $\overline{\Delta}_t := x_t - \hat{x}_t^\circ$. So, when $y_t = C\hat{x}_t^\circ$, ODE (13) should be attended as a differential inclusion. Such *robust adaptive observer* seems to be a more advanced device compared to one containing only a linear (Luenberger type) correction term since it possesses high sensibility within a zone with a small output error. Additionally

$\sigma(\cdot)$ *and* $\varphi(\cdot)$ were selected in the same way that (9). Once all the state is available using the state estimator (13), it is possible to implement the corresponding observed-state $\hat{x}_t$, neural identifier and DNN observer state $\hat{x}_t^\circ$, the admissible control set for $u_t(\widetilde{x}_t)$ should be suggested as an element of the following set

$$U^{adm} := \left\{ u_t := \|u_t\|^2_{\Lambda_u} = \widetilde{\nu}_1 + \widetilde{\nu}_2 \|x_t\|^2_{\Lambda_u^1} + \widetilde{\nu}_3 \left\|\hat{x}_t^\circ\right\|^2_{\Lambda_u^2} \right\} \tag{17}$$

This selection for the admissible control allows the independent design for the controller and the observer (this is referred like a special form of the separation principle). The following theorem introduces the main result on state estimation procedure:

Theorem 2. If there exist positive definite matrices $\Lambda_{\tilde{f}}$, Λ_σ, Λ, Λ_1, $\Lambda_{\tilde{f}}$, Λ_{W_1}, Λ_{W_2}, Λ_{ξ_1}, Λ_{ξ_2}, Λ_u, Λ_D, $\widetilde{Q}_0$, Q_0 and positive constants δ, k, ν_1 such that the matrix Riccati equations (15) have positive definite solutions, then the DNN observer-identifier (11) with any matrix K_1 guarantying that the close-loop matrix $\widetilde{A}^{(0)\bullet}$ is stable, that is, $\widetilde{A}^{(0)\bullet} := \left(A^{(0)\bullet} + K_1 C\right)$ is Hurwitz (this is possible if and only if the pair (C, A) is observable) and $K_2 = \lambda P_1^{-1} C^?, \lambda > 0$ supplied by the learning laws (14) provides the following upper bound for the controlled-state estimation process:

$$\overline{\lim_{T\to\infty}} \frac{1}{T} \int_{t=0}^{T} \|\Delta_t\|_{P_1} dt \le \frac{\rho_s}{\lambda_{\min}\left(P_1^{-1/2} Q_0 P_1^{-1/2}\right)} \tag{18}$$

$$\rho_T := \widetilde{f}_{1,0} + \widetilde{\nu}_1 \left(\Xi + \Xi_1\right) + 4k\sqrt{n} \left\|\xi_{2,t}\right\| + k\nu^{-1} + n\left\|\Lambda_4^{-1}\right\| + 4\Upsilon_2 + \Upsilon_1$$

Bringing together the two stages of control designing, which have been described above, the main result on the *neural- tracking-control* of a class of uncertain non-linear dynamic system subject to state and output external perturbation could be formulated. Due to the non-linear form on the cancer's dynamics, the lack of knowledge about the system structure and the uncertainties presence on the state and output signals, the control function design could be a slow and difficult process (sometimes impossible), specially if any performance index optimization or tracking error minimization are trying to be reached. The differential neural network "modelling" allows defining tracking controllers using the non-linear structure defined by (10). The next subsection gives some general solutions to control non-linear uncertain systems affected by external perturbations.

4.2. Model reference controllers

The control design using the estimated states requires the next definition. Let introduce the performance index describing the tracking quality as:

$$J_t = \lim_{T\to\infty} \sup \int_{t=0}^{T} \left(\left\| (x_t - \hat{x}_t) + (\hat{x}_t - x_t^*) \right\|_Q^2 + \left\| u_t \right\|_R^2 \right) dt \tag{19}$$

Using the inequality $\left\| a+b \right\|_M^2 \le \left(1+\varepsilon^{-1}\right) \left\| a \right\|_M^2 + \left(1+\varepsilon\right) \left\| b \right\|_M^2$ in (19), the performance index could be presented as:

$$J_t \le \left(1+\varepsilon^{-1}\right) \overline{\lim_{T\to\infty}} \int_{t=0}^{T} \left\| (x_t - \hat{x}_t) \right\|_Q^2 dt + \left(1+\varepsilon\right) \overline{\lim_{T\to\infty}} \int_{t=0}^{T} \left(\left\| (\hat{x}_t - x_t^*) \right\|_Q^2 + \left\| u_t \right\|_R^2 \right) dt \tag{20}$$

In view of the last expression, the tracking controller could be described by the minimization (in any sense) of the third term, $\int_{t=0}^{T} \left(\left\| (\hat{x}_t - x_t^*) \right\|_Q^2 + \left\| u_t \right\|_R^2 \right) dt$. The optimization procedure was carried out as follows: Lets consider the non-linear system to be controlled as in (1) and consider, also, a reference model representing the normal conditions for a healthy patient $\dot{x}_t^* = \hat{f}(x_t^*, t)$, x_0^* is fixed where x_t^* is the state of the reference system, $\hat{f}$ is a non-linear function fulfilling the Lipschitz condition with the corresponding constant L^*. If the tracking process is going to be considered, the function $\hat{f}$ could be any which reachable set has a non empty intersection with that one of non-linear function [14], but if the studied case deals with regulation problem, the reference system must be purposed as $f = 0$, $x_0^* = c$, $c \in \Re^n$ is a vector composed by constant elements c_i, $i = \overline{1,n}$. To complete the control design, the control action (u_t) going to be made up by two parts $u_t = \left[W_{2,t}^{\circ} \varphi\left(\hat{x}_t^{\circ}\right) \right]^{+} \left(u_{1,t} + u_{2,t} \right)$ where $u_{1,t} \in \Re^m$ could be described as a direct linearization part, and $u_{2,t} \in \Re^n$ will act as a compensation of the tracking no modeled dynamics. In view of the reference model $\hat{f}(x_t^*, t), x_t^*$ structure, the first section in the control function is selected as: $u_{1,t} = -\left[F_0\left(\hat{x}_t, t\right) - \hat{f}\left(x_t^*, t\right) \right]$ where $F_0\left(\hat{x}_t, t\right)$ is obtained by the inverse model representation given for the DNN structure

$$\begin{aligned} \dot{\hat{x}}_t &= F_0\left(\hat{x}_t, t\right) + F_1\left(\hat{x}_t, t\right) u_t \\ F_0\left(\hat{x}_t, t\right) &= A\hat{x}_t^{\circ} + W_{1,t}^{\circ} \sigma\left(\hat{x}_t^{\circ}\right) + K_1^{\circ}\left(y_t - C\hat{x}_t^{\circ}\right) + K_2^{\circ} \frac{y_t - C\hat{x}_t^{\circ}}{\left\| y_t - C\hat{x}_t^{\circ} \right\|} \\ F_1\left(\hat{x}_t, t\right) &= W_{2,t}^{\circ} \varphi\left(\hat{x}_t^{\circ}\right) \end{aligned} \tag{21}$$

and $u_{2,t}$ will be designed using the, so-called, Local Optimal Control. Unfortunately the weights $W_{1,t}^{\circ}$ and $W_{2,t}^{\circ}$ could be not bounded (at least this fact is unknown before the controller structure is applied). This fact is the main consideration to derive a inverse model control using an auxiliary dynamics $\widetilde{x}_t$ constructed as $\frac{d}{dt}\widetilde{x}_t = \widetilde{F}_0(\widetilde{x}_t,t)+\widetilde{F}_1(\widetilde{x}_t,t)u_t$ where $\widetilde{F}_0(\widetilde{x}_t,t)$ and $\widetilde{F}_1(\widetilde{x}_t,t)$ going to be described below. If this auxiliary dynamic vector converge fast to the DNNO structure, it is possible to use $\widetilde{x}_t$ instead $\hat{x}_t$ in the control development (like in the well known adaptive control based on internal model). Let us define the tracking error between thisas "artificial" system ($\widetilde{x}_t$) and the reference model (x_t^*) as $\delta_t = \widetilde{x}_t - x_t^*$. From the previous definitions, the tracking error time derivative is described by $\dot{\delta}_t = \widetilde{F}_0(\widetilde{x}_t,t)-f(x_t^*,t)+\widetilde{F}_1(\widetilde{x}_t,t)u_t$. Using this control function, the tracking error dynamics is presented as: $\dot{\delta}_t = A\delta_t + u_{2,t} + \tilde{f}$. If $u_{2,t}$ is able to compensate $\tilde{f}$, then the tracking error dynamics is asymptotically stable. The auxiliary dynamics is governed by the differential equation described (following the suggested DNNO structure) as

$$\frac{d}{dt}\widetilde{x}_t = A\widetilde{x}_t + V_{1,t}\sigma(\widetilde{x}_t)+V_{2,t}\varphi(\widetilde{x}_t)u_t + \overline{K}_1\Delta_t + \overline{K}_2 SIGN(\Delta_t) \tag{22}$$

where $\Delta_t := \hat{x}_t^{\circ} - \widetilde{x}_t$, $\widetilde{x}_t, \sigma(\cdot) \in \Re^n$, $\varphi(\widetilde{x}_t), V_{1,t}, V_{2,t}, K_1, K_2 \in \Re^{n\times n}$. The parameters involved in this scheme are adjusted using the following matrix differential equations

$$\frac{d}{dt}\widetilde{V}_{j,t} = -k_{V_j}\left(P_2\widetilde{\Delta}_t^T + P_3\widetilde{x}_t^T\right)\overline{\chi}_j^T; \quad \overline{\chi}_1 := \sigma(\widetilde{x}_t), \overline{\chi}_2 := \varphi(\widetilde{x}_t)u_t, k_{V_j} > 0 \tag{23}$$

where P_2 and P_3 are the positive definite solution of the second Riccati equation in (13) and the following one $P_3\widetilde{A} + \widetilde{A}^T P + PR_3P + Q_3 = 0$. It easy to proof that $\hat{x}_t^{\circ} \sim \widetilde{x}_t$ has a finite upper bound for the auxiliary error Δ_t. So the controller design could be abstracted by the following:

Theorem 3. *If the control function u_t^* is proposed as*

$$u_t = [F_1(\hat{x}_t,t)]^+\left(u_{0,t} + u_{1,t}\right), \quad u_{0,t} := F_0(\hat{x}_t,t) - \varphi\left(t, x_t^*\right) \tag{24}$$

and if $u_{1,t}$ is suggested as $u_{1,t} = -2R^{-1}P_c\delta_t$ where P_c is solution for the

Lyapunov function , *then* $\|\delta_t\|_{\bar{Q}}^2 + \|u_{2,t}\|_R^2 \le \|\tilde{f}\|_{A^T Q A^{-1}}^2$ *where the semi-norm* $s_t^2{}_{\Pi}$ *is defined by* $\limsup_{T\to\infty} \frac{1}{T}\int_0^T s_t^T \Pi s_t dt$.

Proof . The proof of this theorem is going to be described in [6].

5. Numerical Results

The cancer term corresponds more than one hundred of different illness. Each of they have particular characteristics, being able to consider itself independent illness, with their causes, evolution and specific treatment. The specific method of treatment used is determined by the cancer's type, stage, and location. There exist several treatments for this disease: chemotherapy, radiotherapy, surgery, etc. Although development of new technics of surgery and transplants, more effective drugs and better radiation methods, often some malignant cells survives to these therapies and disseminates into the organism (the so-called metastasis process). Recently, treatment efforts implementing immunotherapy are being investigated [15]. Although, the immunotherapy dynamics is difficult to be modeled (like many other biological processes), because it has many uncertainties in its description, it depends strongly on the input function, etc. One common solution for this problem is to consider this model as a black box, whereas only input and output information is supposed to be measurable. There are some theoretical studies that have been forcing into the order to modelling this particular cancer treatment method [16] The cancer model is based on the immunotherapy response describing the activated immune or effector cells (x_t), the tumor cells (y_t) and *IL-2* concentration (z_t) in the single tumor-site compartment

$$\begin{aligned}
\dot{x}_t &= cy_t - \mu_2 x_t + \frac{p_1 x_t z_t}{g_1 + z_t} + u_t s_1; \\
\dot{y}_t &= r_2 y_t (1 - b y_t) - \frac{a x_t y_t}{g_2 + y_t}; \\
\dot{z}_t &= \frac{p_2 x_t y_t}{g_3 + y_t} - \mu_3 z_t
\end{aligned} \tag{24}$$

Obviously the main result is devoted to adjust this function in order to reduce the tumour cells, considering the physiological restrictions given by the each patient under immunotherapy treatment.

5.1. Control Process results with incomplete information

The first stage in the controller developed is the DNN training process. This is carried out using some pattern data that usually are obtained using some real experiments or a database which contains some particular information about the nonlinear system to be controlled. Unfortunately, there are not on-line sensors that could measure any variable

related with the immunotherapy treatment. As far as the authors known, there are not a database which contains the time evolution of the immunotherapy dynamics. That is the main reason to use an artificial mathematical model of the IL-2 interaction in patients which are assisted with immunotherapy, as a data generator to be used in the rest of the document. To guarantee the estimation error convergence "close" (depending on the perturbations values) to zero, training process for the DNN should be solved in order to obtain the best possible values for the free-design parameters in the identification design are adjusted using the learning laws described above. This adjustment procedure starts with the data set testing for the identifier structure. The data set is obtained using the mathematical description with a fixed initial condition. So if a bad performance in the error convergence is obtained, a new set of matrices is used to test again the identifier, and so on. Besides, the neural networks weights are adjusted using two matrix differential equations, so it is important the initial weights values selection.

The control problem for immunotherapy dosage is difficult because the designed controller should be restricted by real patient settings. In view of this, it is difficult to suggest any performance index to solve the corresponding optimization problem. In real situations, the health improvement for the patient is slow and progressive, so the controller must consider this important medical fact. Both elements recently mentioned implies the reference model should be designed in order to represent a slightly healthy condition for the patient. In order to solve the numerical example to evaluate the capabilities for this method, a modified parameter set for the DNNO corresponding to the healthy patient has been suggested [16]. The adaptive controller was designed using the local optimal control. Taking into account the obtained data in the estimation procedure; it is possible to apply the adaptive control function designed to make non-linear system dynamics (1) follows the corresponding reference trajectory. This sub-optimum controllers does not requires the trial and error method to obtain the previous matrix value. It is solved analytically which ensures there is positive solution for the Lyapunov equation selecting the implicated eigenvalues at $\lambda_1 = 2.56 \times 10^{-2}$, $\lambda_2 = 6.34 \times 10^{-2}$, $\lambda_3 = 3.42 \times 10^{-2}$. The next figure shows the performance for this observer based-controller considering there is not any mathematical model (this is true in principle because the same procedure using on-line information or a database containing such knowledge). It is important to note the total amount of tumour cells has a significative reduction once the controller was applied (upper right corner). However the effectors cells are not so affected by the immunotherapy dosage (upper left corner in). Another important fact in this controlled process is the important reduction on the IL-2 concentration in the body (lower left corner) diminishing a possible intoxication possibility. The control process quality could be evaluated using the performance index given by (), where the convergence between the immunotherapy model and the reference one is displayed. This element can be evaluated directly from the tracking figures. This approach could be used like a dosage predictor which is able to modify the immunotherapy strategy, this can be done using a computer with a friendly software which is ready to receive the current data on the IL-2 concentration and then to suggest the corresponding application period, i.e. if the dosage should be supplied or not. The last element in the previous figure shows the reference input that has

demonstrated the best possible control action to diminish the cancer effects, and the control function generated by the DNN adaptive controller.

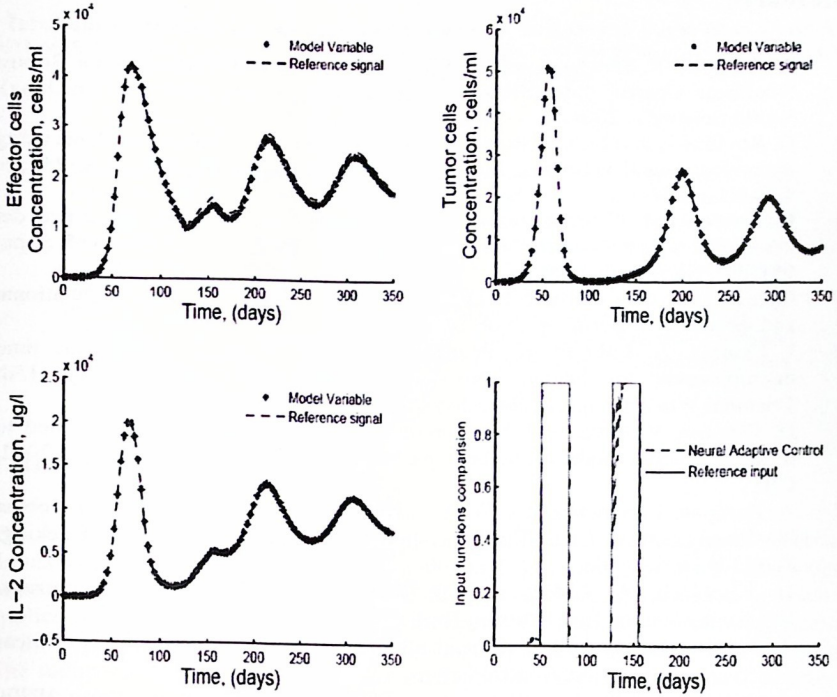


Figure 1. Adaptive Neural Control Process results. The figure at the upper right corner shows the effector cells tracking performance. It is important to note the close relationship between both variables. The same results can be attained for tummor cells and IL-2 concentrations. The last graph demonstrated the reference input (suggetsted by physician) estimation given by the DNNC.

6. Conclusions

In this paper a new model-free neural control is suggested and analyzed. It consists in robust approach application to control a simplified model of a differential neural network observer that represents a model of an uncertain non-linear system to be controlled. The upper bound for the averaged tracking error is established if the local optimal neural control be applied. The control of the real immunotherapy process for cancer treatment is considered. These results suggested the possible application in real

medical procedures.

References

[1] A. Poznyak, E. Sanchez, and W. Yu, Differential Neural Networks for Robust Nonlinear Control (Identification, state Estimation an trajectory Tracking). World Scientific, 2001.

[2] G. Rovithakis and M. Christodoulou, Adaptive control of unknown plants using dynamical neural networks, IEEE Trans. Systs.,Man and Cyben., vol. 24, pp. 400-412, 1994.

[3] H. Fonseca, H. Cabrera, A.and Dominguez, and G. Ramirez, Dinámica del modelo de bergman controloda por un sistema neurodifuso, IFMBE, vol. 5, pp. 951-954, September 2004.

[4] Identification of a fedbatch fermetation process: comparision of computational and laboratory experiments, vol. 24, 2002.

[5] I. Chairez, A. Cabrera, K. Poznyak, and T. Poznyak, A continuous time neuroobserver for human immunodeficiency virus (hiv) dynamics, 15th Triennial World Congress, Barcelona, Spain,, 2002. IFAC.

[6] H. Fonseca, V. Ortiz, and A. Cabrera, Stochastic neural networks applied to dynamic glucose model for diabetic patients, Acapulco Guerrero, Mexico, 2002. CIE.

[7] A. Poznyak, E. Sanchez, and W. Yu, Differential Neural Networks for Robust Nonlinear Control (Identification, State Estimation an Trajectory Tracking. World Scietific, 2001.

[8] H. Knobloch, A. Isidori, and D. FLocherzi, Topics in control therory, Birkhauser Verlag, Basel-Boston-Berlin, 1993.

[9] A. Krener and A. Isidori, Linealization by output injection and non-linear observers, Systems and Control Letters, vol. 3, pp. 47-52, 1983.

[10] A. H. Jazwinski, Stochastic Process and filtering theory. Academic Press, 1970.

[11] T. Poznyak and A. Chairez, I. Poznyak, Application of a neural observer to phenols ozonation in water: Simulation and kinetic parameters identification," Water Research, vol. 39, pp. 2611-2620, 2005.

[12] L. Lung, System Identification, Theory for the user. Springer-Verlag, 1979.

[13] D. Luenberger, Obseving the state of liinear sistems, IEEE transactions on Military Electronics, vol. 8, pp. 74-90, 1964.

[14] A. B. Kurshanski and P. Varaiya, Ellipsoidal techniques for reachability analysis, Hybrid Systems: Computation and Control, vol. 1790, pp. 202-214, 2000. Lecture Notes in Computer Science, New York, New York.

[15] S. Muñoz and C. Pedemonte, Inmunoterapia: mecanismos de acción, indicaciones y beneficios, Asociacion Española de Pediatria, p. 127, 2003.

[16] T. Burden, J. Ernstberger, and K. Fister, Optimal control applied to immunotherapy, Discrete and Continuos Dymanical Systems, vol. 4, pp. 135-146, February 2004.

A Direct Torque Vector Neural Control of a Three Phase Induction Motor

Ieroham S. Baruch, Carlos-Roman Mariaca G., and Irving-Pavel de la Cruz A.
CINVESTAV-IPN, Department of Automatic Control, Av. IPN No 2508, Col. Zacatenco, A.P. 14-740, 07360 Mexico D.F., Mexico
{baruch, cmariaca}@ctrl.cinvestav.mx

(Paper received on August 11, 2006, accepted on September 26, 2006)

Abstract. The paper proposed a neural solution to the direct torque vector control of three phase induction motor including real-time trained recurrent neural velocity controller and a hysteresis flux and torque controllers, which permitted the speed up reaction to the variable load. The basic equations and elements of the direct field oriented torque control scheme are given. The control scheme is realized by one RNN learned by a real-time BP algorithm and three FFNNs learned off-line by Levenberg-Marquardt algorithm with data taken from PI-control scheme 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 Neural Networks (NN) applications for identification and control of electrical drives became very popular in last decade. In [1], a multilayer-perceptron-based-neural-control is applied for a DC motor drive. 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, particularly the 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, AC signals processing, and 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]-[6]. There exist two methods for PWM current controlled inverter – direct and indirect vector control, [3]. This paper considered 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 [7] a Feedforward NN (FFNN) is used for vector PW modulation, resulting in a faster response. In [8] an Artificial NN is used for fast estimation of the angle used in a FOC system. Some basic principles of the direct torque control of IM drives are given in [9]. The [10] applied NNs in the direct torque control of IM drive. In [11], a FFNN is used for commutation table emulation in a direct torque control of IM. In [12], the authors proposed to use a NN so to compensate the variations of the

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 131-140

stator resistance, necessary for the FOC of flux in a direct torque IM control scheme. In [13] off-line trained FFNNs-are used to substitute the blocks of coordinate transformation, vector flux computation, torque estimation, and commutations table realization in a direct self-tuning IM control scheme. In [14] a FFNN-based estimator of the feedback signals is used for induction motor drive direct FOC system. The paper [15] proposed two NN-based methods (direct and indirect) for FOC of induction motors. The results and particular solutions obtained in the referenced papers shows that the application of NN offers a fast and improved alternative of the classical FOC schemes. The present paper proposed a neural solution of a direct torque vector control problem that assures fast response and adaptation to a variable load.

2. Mathematical Models of the Three Phase Induction Motor

A Phase (a, b, c) Model: The Induction Motor (IM) equations, [5], [6], for stator and rotor voltages in vector-matrix form are given as:

$$\mathbf{v}_{abcs} = \mathbf{r}_s \mathbf{i}_{abcs} + p\boldsymbol{\lambda}_{abcs}; \mathbf{v}_{abcr} = \mathbf{r}_r \mathbf{i}_{abcr} + p\boldsymbol{\lambda}_{abcr}; \mathbf{r}_s = r_s \mathbf{I}_3; \mathbf{r}_r = r_r \mathbf{I}_3 \tag{1}$$

$$\begin{aligned} &\mathbf{v}_{abcs} = (v_{as}, v_{bs}, v_{cs})^T; \mathbf{v}_{abcr} = (v_{ar}, v_{br}, v_{cr})^T \\ &\mathbf{i}_{abcs} = (i_{as}, i_{bs}, i_{cs})^T; \mathbf{i}_{abcr} = (i_{ar}, i_{br}, i_{cr})^T \\ &\boldsymbol{\lambda}_{abcs} = (\lambda_{as}, \lambda_{bs}, \lambda_{cs})^T; \boldsymbol{\lambda}_{abcr} = (\lambda_{ar}, \lambda_{br}, \lambda_{cr})^T \end{aligned} \tag{2}$$

Where the variables are: voltage, current, and flux, stator and rotor, three dimensional (a, b, c) vectors, with given up phase components; $\mathbf{r}_s$ and $\mathbf{r}_r$ are stator and rotor winding resistance diagonal matrices, with given up equal elements r_s and r_r, respectively; $\mathbf{I}_3$ is an identity matrix with dimension three, and p is a Laplacian differential operator. The vector-matrix block-form representation of the flux leakage is given by the equation:

$$\begin{bmatrix} \boldsymbol{\lambda}_{abcs} \\ \boldsymbol{\lambda}_{abcr}^{'} \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{ss}^{abc} & \mathbf{L}_{sr}^{'abc} \\ (\mathbf{L}_{sr}^{'abc})^T & \mathbf{L}_{rr}^{'abc} \end{bmatrix} \begin{bmatrix} \mathbf{i}_{abcs} \\ \mathbf{i}_{abcr}^{'} \end{bmatrix} \tag{3}$$

Where: the stator, rotor and mutual block-inductance (3x3) matrices are described in [5], [6]. The relative leakage inductance depends on the winding turn stator/rotor ratio n, and on the angular rotor position $_r$ respectively [5], [6]. Finally, the voltage equations (1) could be expressed with respect to the stator in the (a, b, c) model form:

$$\begin{bmatrix} \mathbf{v}_{abcs} \\ \mathbf{v}_{abcr}^{'} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_s + p\mathbf{L}_{ss}^{abc} & p\mathbf{L}_{sr}^{'abc} \\ (p\mathbf{L}_{sr}^{'abc})^T & \mathbf{r}_r + p\mathbf{L}_{rr}^{'abc} \end{bmatrix} \begin{bmatrix} \mathbf{i}_{abcs} \\ \mathbf{i}_{abcr}^{'} \end{bmatrix}; \begin{aligned} &\mathbf{v}_{abcr}^{'} = n\mathbf{v}_{abcr}; \mathbf{i}_{abcr}^{'} = (1/n)\mathbf{i}_{abcr} \\ &\boldsymbol{\lambda}_{abcr}^{'} = n\boldsymbol{\lambda}_{abcr}; \mathbf{r}_r^{'} = n^2 \mathbf{r}_r \end{aligned} \tag{4}$$

Where the given up variables are the relative rotor voltage, the current, the flux and the resistance.

A (q, d, 0) Model: The (a, b, c) model is very complicated for control, so it could be simplified using a transformation to the (q, d, 0) form. The AC motor equations for the stator and rotor voltages in (q, d, 0) vector-matrix form are given as follows:

$$\mathbf{v}_{qd0s} = \mathbf{r}_s \mathbf{i}_{qd0s} + \Omega\boldsymbol{\lambda}_{qd0s} + p\boldsymbol{\lambda}_{qd0s}\,;\; \mathbf{v}'_{qd0r} = \mathbf{r}_r \mathbf{i}'_{qd0r} + \Delta\Omega\boldsymbol{\lambda}'_{qd0r} + p\boldsymbol{\lambda}'_{qd0r} \tag{5}$$

$$\begin{aligned}
&\mathbf{v}_{qd0s} = \left(v_{qs}, v_{ds}, v_{0s}\right)^T ; \mathbf{v}'_{qd0r} = \left(v'_{qr}, v'_{dr}, v'_{0r}\right)^T \\
&\mathbf{i}_{qd0s} = \left(i_{qs}, i_{ds}, i_{0s}\right)^T ; \mathbf{i}'_{qd0r} = \left(i'_{qr}, i'_{dr}, i'_{0r}\right)^T \\
&\boldsymbol{\lambda}_{qd0s} = \left(\lambda_{qs}, \lambda_{ds}, \lambda_{0s}\right)^T ; \boldsymbol{\lambda}'_{qd0r} = \left(\lambda'_{qr}, \lambda'_{dr}, \lambda'_{0r}\right)^T
\end{aligned} \tag{6}$$

Where the variables are: voltage, current, and flux, stator and rotor, three dimensional vectors, with given up components; $\mathbf{r}_s$ and $\mathbf{r}_r$ are stator and rotor resistance diagonal matrices, given by (1); , and are diagonal angular velocity matrices, given by:

$$\Omega = \begin{bmatrix} \omega & 0 & 0 \\ 0 & -\omega & 0 \\ 0 & 0 & 0 \end{bmatrix}; \Delta\Omega = \begin{bmatrix} \omega - \omega_r & 0 & 0 \\ 0 & -(\omega - \omega_r) & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{7}$$

The vector-matrix block-form representation of the flux leakage is given by:

$$\begin{bmatrix} \boldsymbol{\lambda}_{qd0s} \\ \boldsymbol{\lambda}'_{qd0r} \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{ss}^{qd0} & \mathbf{L}_{sr}^{qd0} \\ (\mathbf{L}_{sr}^{qd0})^T & \mathbf{L}_{rr}^{qd0} \end{bmatrix} \begin{bmatrix} \mathbf{i}_{qd0s} \\ \mathbf{i}'_{qd0r} \end{bmatrix} \tag{8}$$

Where: the stator, rotor and mutual block-inductance (3x3) matrices are given in [5], [6]. The (q, d, 0) model could be written in the stationary and synchronous frames taking the angular velocity equal to: =0 and = $_e$ where $_e$, corresponds to the angular velocity of the stator field. Now we could write the scalar electromagnetic torque equation that could be expressed in the following basic forms used:

$$\begin{aligned}
&T_{em} = \frac{3}{2}\frac{P}{2}\boldsymbol{\lambda}'^{T}_{q-d,r}\mathbf{i}'_{d-q,r} = \frac{3}{2}\frac{P}{2}\mathbf{i}'^{T}_{d-q,r}\mathbf{i}_{q-d,s}\,; \\
&\mathbf{i}_{q-d,s} = \left(i_{qs}, i_{ds}\right)^T ; \mathbf{i}'_{d-q,r} = \left(i'_{dr}, -i'_{qr}\right)^T ; \boldsymbol{\lambda}'_{q-d,r} = \left(\lambda'_{qr}, \lambda'_{dr}\right)^T
\end{aligned} \tag{9}$$

where: P is a number of poles.

Field Orientation Conditions: The flux and torque equations 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, [6]. 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 ($'^{e}_{dr} = {}_{r}$), 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:

$$\lambda'^{e}_{qr} = 0; p\lambda'^{e}_{qr} = 0; \lambda_r = \lambda'^{e}_{dr} \tag{10}$$

Taking into account that the rotor windings are shortcut, (the rotor voltage is zero) and the given up field orientation conditions the first two components of (5) become:

$$0 = r'_r i'^{e}_{qr} + (\omega_e - \omega_r)\lambda'^{e}_{dr}; 0 = r'_r i'^{e}_{dr} + p\lambda'^{e}_{dr} \tag{11}$$

From (5), for the q-component of the rotor flux, it is obtained:

$$\lambda'^{e}_{qr} = L_m i^{e}_{qs} + L'_r i'^{e}_{qr} = 0; L'_r = L'_{lr} + L_m; i'^{e}_{qr} = -(L_m / L'_r) i^{e}_{qs}; \tag{12}$$

Taking into account (10) and (12), the torque equation (9) obtained the final form:

$$T_{em} = \frac{3}{2}\frac{P}{2}\frac{L_m}{L'_r}\lambda'^{e}_{dr} i^{e}_{qs} \tag{13}$$

The equation (13) 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 (11), taking into account (12) it is easy to obtain the slipping angular velocity as:

$$\omega_e - \omega_r = (r'_r L_m / L'_r)(i^{e}_{qs} / \lambda'^{e}_{dr}) \tag{14}$$

The final equations (12), (13), (14) gives us the necessary basis for a direct decoupled field oriented (vector) control of the AC motor drive, where following Fig. 1(b), the q-component of the stator current produced torque and the d-component of the stator current produced flux.

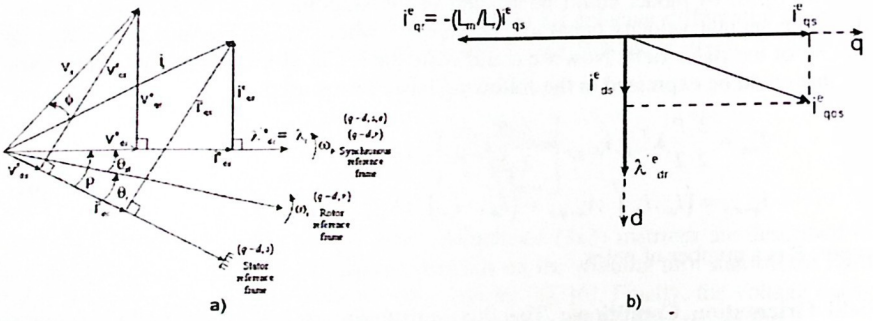


Fig. 1. Vector diagrams of the stator current, voltage and the rotor flux. a) The current and voltage vector representations in stator and in rotor synchronous reference frames. b) The stator current and the rotor flux vector representations in synchronous reference frame.

Coordinate Transformations: The combined stator current transformation from (a, b, c) to (q-d,s,e) synchronous reference frame [5], [6], is given by the equation:

$$\begin{bmatrix} i^e_{qs} \\ i^e_{ds} \end{bmatrix} = \begin{bmatrix} \cos\rho & f_1 & f_2 \\ sin\rho & f_3 & f_4 \end{bmatrix} \begin{bmatrix} i_{as} \\ i_{bs} \\ i_{cs} \end{bmatrix}; \quad \begin{aligned} f_1 &= [-(1/2)\cos\rho - (\sqrt{3}/2)sin\rho] \\ f_2 &= [-(1/2)\cos\rho + (\sqrt{3}/2)sin\rho] \\ f_3 &= [-(1/2)sin\rho + (\sqrt{3}/2)\cos\rho] \\ f_4 &= [-(1/2)sin\rho - (\sqrt{3}/2)\cos\rho] \end{aligned} \tag{15}$$

Flux and Torque Estimation: From the equation (5), written for the stationary reference frame (=0), we could obtain:

$$\lambda^s_{qs} = (1/p)\left(v^s_{qs} - r_s i^s_{qs}\right); \lambda^s_{ds} = (1/p)\left(v^s_{ds} - r_s i^s_{ds}\right) \tag{16}$$

The stator flux part of the equation (8) could be resolved for the rotor currents, as it is:

$$i^{'s}_{qr} = (\lambda^s_{qs} - L_s i^s_{qs})/L_m; i^{'s}_{dr} = (\lambda^s_{ds} - L_s i^s_{ds})/L_m \tag{17}$$

Substituting (17) back in the rotor flux part of (8) we could obtain:

$$\lambda^{'s}_{qr} = (L'_r/L_m)\left(\lambda^s_{qs} - L'_s i^s_{qs}\right); \lambda^{'s}_{dr} = (L'_r/L_m)\left(\lambda^s_{ds} - L'_s i^s_{ds}\right); L'_s = [L_s - (L^2_m/L'_r)] \tag{18}$$

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 is:

$$\begin{aligned} &\lambda'_r = \sqrt{(\lambda^{'s}_{qr})^2 + (\lambda^{'s}_{dr})^2}; \rho = \tan^{-1}(\lambda^{'s}_{qr}/\lambda^{'s}_{dr}); \\ &sin\rho = \lambda^{'s}_{qr}/\lambda'_r; \cos\rho = \lambda^{'s}_{dr}/\lambda'_r \end{aligned} \tag{19}$$

The torque equation (13) could be rewritten in the form:

$$T_{em} = \frac{3}{2}\frac{P}{2}\frac{L_m}{L'_r}\lambda'_r i^e_{qs} \tag{20}$$

3. Direct Torque vector Control of the Induction Motor

A General Control Scheme: A general block diagram of the direct torque vector control of the Induction Motor drive is given on Fig. 2(a). The direct torque control scheme contains three principal blocks. They are: G1 – block of a velocity PI controller; G2, G3 –blocks of hysteresis flux and torque controllers; B1 – block of angle (N) computation; block of voltage commutations look up table; block of coordinate (a, b, c) to (q-d, s, e) transformation (see equation. (15)); block of a vector estimation, performing the field orientation, i.e. the torque, the flux and the angle- -computations, see equations (30), (31); block of the IM converter-machine system. The block of the IM

converter machine system contains a three phase bridge ASCI DC-AC voltage fed inverter, an induction motor model, and a model of the whole mechanical system driven by the IM ((2/P)J(d ω/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, so it contains various blocks, illustrated by Figs. 2(b) and 2(c). Fig. 2(b) illustrates the flux and angle estimation for field orientation, computing (16), (18), (19). The rotor flux computations block (see Fig. 2(b)) performs computations given by (16), (18), illustrated by Fig. 2(c). The direct torque control is effectuated through a look up table of voltage commutations that determine the on-off position of the thyristors in three phase bridge inverter. The table entries are the torque and flux errors and the stator field angle. The torque and flux errors are discretized in 60° phase angle by hysteresis comparators of two and three levels, respectively.

A Neural Networks Realization of the Control Scheme: The simplified Block-Diagram (BD) of the indirect neural vector control system, given on Fig. 3 is partly realized by four NNs, which function, topology and learning are briefly described.

The first Recurrent NN1 (RNN1) is an angular velocity recurrent neural controller with one input (the velocity error), one output (the torque set point), where the To = 0.01 sec. is chosen with respect to the stator electrical time constant. The weights learning is done in real – time using the Backpropagation (BP) algorithm. The RNN topology and learning are described in [2]. The RNN1 function is given by:

$$T^*(k+1) = \varphi\{c(k+1)\cdot\varphi[-a(k)x(k)+b(k)e_{vel}(k)-w^h(k)]-w^o(k+1)\} \qquad (21)$$

Where: a(.) and b(.) are hidden layer RNN1 weights; c(.) is an output layer RNN1 weight; w^h(.), w^o(.) are threshold weights of the hidden and output RNN1 layers, respectively; is a tanh activation function; e_{vel} is a velocity error; T* is the torque set point – output of the RNN1. The BP algorithm of learning for the output layer of the RNN1, [2], is given by:

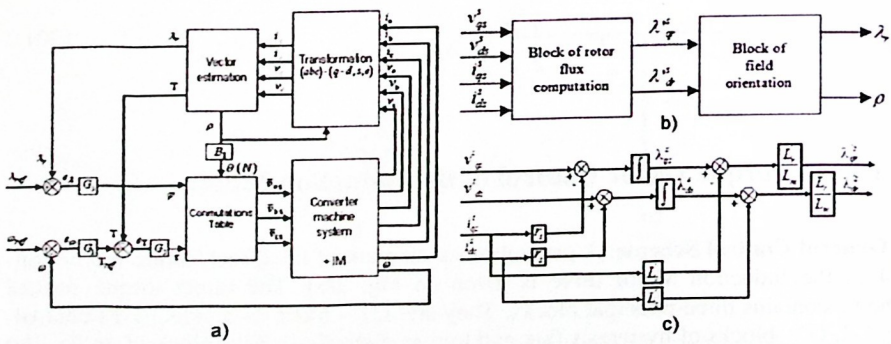


Fig. 2. Block diagrams. a) General BD of a direct IM vector control. b) BD of the vector estimation computations. c) BD of the flux estimation computations.

$$c(k+1) = c(k) + \eta e_{wf}(k)[1-(T^*(k))^2]z(k); \qquad (22)$$
$$w^n(k+1) = w^n(k) + \eta e_{wf}(k)[1-(T^*(k))^2](-1)$$

The BP algorithm of learning for the hidden layer of the RNN1, [2], is given by:

$$a(k+1) = a(k) + \eta R(k)x(k); b(k+1) = b(k) + \eta R(k)e_{wf}(k) \qquad (23)$$
$$w^h(k+1) = w^h(k) + \eta R(k)(-1); R(k) = c(k)e_{wf}(k)[1-(z(k))^2]$$

Where the learning rate is: =0.01.

The second Feedforward NN2 performed the rotor torque estimation (see equation (20)). The rotor (q-d,r) flux components $^{s}_{qs}$, $^{s}_{ds}$ are previously computed using equation (16) (see Fig.2c), and they are inputs of FFNN2. The other two inputs are the stator current components- i^{s}_{qs}, i^{s}_{ds}. The FFNN2 output is the torque- T_{em}. The FFNN2 topology is (4-20-5-1). The FFNN2 learning is off-line, applying the Levenberg-Marquardt (L-M) algorithm [16], [17]. The final value of the MSE reached is of 10^{-10}. The FFNN2 is learned and generalized by 2500 input-output patterns in 1179 epochs.

The third Feedforward NN3 performed rotor flux estimation using (19) equation. The rotor (q-d,r) flux components- $^{s}_{qs}$, $^{s}_{ds}$ are previously computed using equation (16) (see Fig.2c), and they are inputs of FFNN3. The FFNN3 output is the rotor flux- $_{r}$. The FFNN3 topology is (2-20-5-1). The FFNN3 learning is off-line, applying the L-M algorithm. The final value of the MSE reached is of 10^{-10}. The FFNN3 is learned and generalized by 2500 input-output patterns in 617 epochs.

The fourth Feedforward NN4 performed rotor flux estimation using (19) equation. The rotor (q-d,r) flux components- $^{s}_{qs}$, $^{s}_{ds}$ are previously computed using equation (16) (see Fig. 2©), and they are inputs of FFNN4. The FFNN4 output is the field angle . The FFNN4 topology is (2-20-5-1). The FFNN4 learning is off-line, applying the L-M algorithm. The final value of the MSE reached is of 10^{-10}. The FFNN4 is learned and generalized by 2500 input-output patterns in 727 epochs.

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. Figs. 3 (a) and 3 (b) show the angular velocity set point vs. the IM angular velocity in the general case of velocity control and particularly with load torque changes. The results show that the angular velocity control system has a fast speed up response and satisfactory behaviour in the case of load change. Figs. 4(a) and 4 (b) show the flux graphics of control system with hysteresis control applying the classical control scheme and applying the scheme using NNs. The results show a faster and better re-

sponse of the neural system. Figs. 5(a) and 5(b), Figs. 6(a) and 6(b) show the torque graphics with hysteresis control in the same cases and a load changes. The results show a faster and better response of the neural system. Figs. 7(a) and 7(b) and Figs. 7(c) and 7(d) shows the (a,b,c) stator currents of hysteresis controlled system using classical and neural control schemes in load changes conditions for different time intervals. The results show a good performance of the neural control system at all.

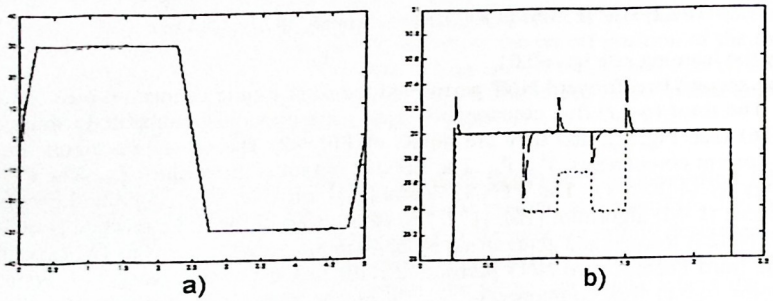


Fig. 3. Graphical results of the IM velocity control. a) General graphics of the angular velocity control; b) Graphical results of angular velocity control with load changes.

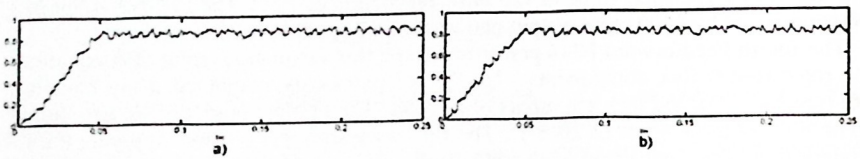


Fig. 4. Graphical results of the IM flux control. a) Graphics of the flux classical control; b) Graphics of the flux control using neural networks.

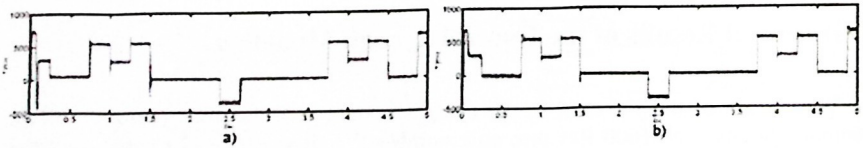
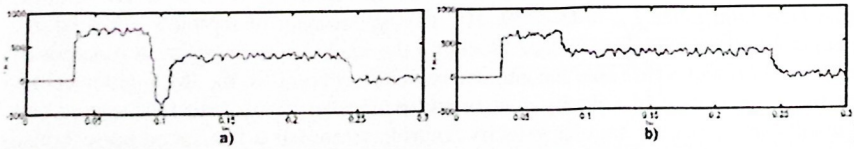


Fig. 5. Graphics of the process history of the torque control with load changes. a) Graphics of the torque classical control. b) Graphics of the torque control using neural networks.



g. 6. Detailed graphics of the torque control using both control schemes and load changes. a) Classical control; b) Control using neural networks.

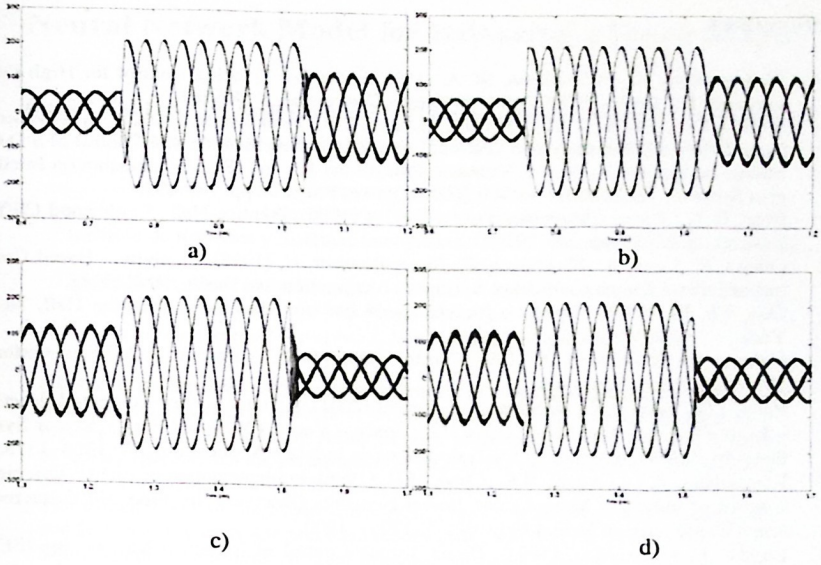


Fig. 7. Graphical results of (a,b,c) stator currents during load changes. a) and c) Classical control. b) and d) Control using neural networks.

5. Conclusions

The paper proposed a neural network solution to the direct torque vector control of a three-phase induction motor including velocity RNN controller, and hysteresis controllers for flux and torque, which permitted the speed up reaction to the variable load. The period of discretization of the RNN velocity controller is chosen with respect to the AC stator electrical time constant. The basic equations and elements of the direct field oriented torque control scheme are given. The control scheme is realized by one RNN BP real-time velocity controller and three FFNN learned off-line by Levenberg-Marquardt algorithm with data taken by classical control simulations. The LM algorithm has a 10^{-10} set up error precision. The graphical simulation results exhibited a better performance of the adaptive NN control system with respect to the classical control in variable load conditions.

Acknowledgements

We want to thank CONACYT, Mexico for the scholarships given to PhD student Carlos Roman Mariaca Gaspar and MS student Irving Pavel de la Cruz Arguello, during his studies in CINVESTAV-IPN, Mexico.

References

[1] Weerasooriya, S., El-Sharkawi, M. A. (1991): Adaptive Tracking Control for High Performance DC Drives. IEEE Trans. on Energy Conversion, 4:182-201.

[2] Baruch, I. S., Flores, J. M., Nava, F., Ramirez, I. R., Nenkova, B. (2002). 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, pp. 289-295.

[3] Bose, B. K.: Power Electronics and AC Drives (1996). Prentice-Hall , Englewood Cliffs, New Jersey 07632, pp. 264-291.

[4] Ortega, R., Loria, A., Nicklasson, P., Sira – Ramirez, H. (1998). Passivity – Based Control of Euler – Lagrange Systems. Springer – Verlag, London, Berlin, Heidelberg.

[5] Ong, Ch. M. (1998). Dynamic Simulation of Electric Machinery. Prentice Hall, New York.

[6] Novotny, D. W., Lipo, T. A. (1996). Vector Control and Dynamics of AC Drives. Oxford University Press, New York.

[7] Pinto, J.O., Bose, B.K., da Silva, L. E. Borges (2001). A Stator – Flux – Oriented Vector – Controlled Induction Motor Drive With Space – Vector PWM and Flux – Vector Synthesis by Neural Networks. IEEE Transactions on Industry Applications, 37: 1308–1318.

[8] Keerthipala, W.L., Duggal, B.R., Chun, M.H. (1996). Implementation of Field – Oriented Control of Induction Motors using Neural Networks Observers. In: Proc. IEEE International Conference on Neural Networks, 3: 1795 – 1800.

[9] Ludtke, I., Jayne, M.G. (1995). Direct Torque Control of Induction Motors. In: IEEE Colloquium on Vector Control and Direct Torque Control of Induction Motors, London, U.K.

[10] Cabrera, L.A., Elbalak, M.E., Zinder, D.S. (1997). Learning Techniques to Train Neural Networks as a State Selector for Inverter Fed Induction Machines Using Direct Torque Control. IEEE Transactions on Power Electronics, 12:788 – 799.

[11] Orille, A. L., Sowilam, G. M. A. (1999). Application of Neural Networks for Direct Torque Control, Computers and Industrial Engineering, 37:391-394.

[12] Cabrera, L.A., Elbalak, M.E., Husaim, I. (1997). Tuning the Stators Resistance of Induction Motors Using Artificial Neural Networks. IEEE Transactions on Power Electronics, 12:779–787.

[13] Shi, K.L., Chan, T.F., Wong, Y.K., Ho, S.L. (2001). Direct Self Control of Induction Motor Based on Neural Network. IEEE Transactions on Industry Applications, 37:1290–1298.

[14] Simoes, M. G., Bose, B. K. (1995). Neural Network Based Estimation of Feedback Signals for a Vector Controlled Induction Motor Drive. IEEE Transactions on Industry Applications, 31:620–629.

[15] Ba – Razzouk, A., Cheriti, A., Olivier, G., Sicard, P. (1997). Field – Oriented Control of Induction Motors Using Neural – Network Decouplers. IEEE Transactions on Power Electronics, 12:752-763.

[16] Hagan, M. T., Menhaj, M. B. (1994). Training Feedforward Networks with the Marquardt Algorithm. IEEE Transactions on Neural Networks, 5:989-993.

[17] Demuth, H., Beale, M. (1992-2002). Neural Network Toolbox User's Guide, version 4, The Math Works, Inc. COPYRIGHT.

Neural Network Model for Balancing a Biped Robot

Daniel Zaldívar [1,2], Erik Cuevas [1,2], Marco A. Pérez-Cisneros[2], Raúl Rojas [1]
[1] Freie Universität Berlin, Institüt für Informatik, Takustr. 9, D-14195
Berlin, Germany
{zaldivar, cuevas, rojas}@inf.fu-berlin.de
http://www.inf.fu-berlin.de/~zaldivar
[2] Instituto de Robótica y Sistemas Inteligentes, CUCEI, Universidad de Guadalajara
marcopc@cucei.udg.mx
Blvd. Marcelino García Barragan 1421, C.P. 44430, Guadalajara, Jal., México
http://irsi.cucei.udg.mx
(Paper received on August 12, 2006, accepted on September 27, 2006)

Abstract. In bipedal walking, stable balance and walking sequence are essential. In this work, one neural network is proposed to model the balance dynamic of a biped robot. The generalizing ability of back propagation neural network is used to agilely characterize the performance of a fuzzy PD incremental algorithm based on the Zero Moment Point (ZMP) criteria to balance a real biped robot structure. The effectiveness of the implemented neural model is demonstrated by comparison between its output -the predicted robot's ZMP, and the real robot's ZMP value. Some training algorithms are used to model the biped balance and its results are reported.

1. Introduction

In recent years, there is enthusiasm to study the bipedal walking as private companies such as Sony, Honda, etc., alongside other research institutes and universities have invested huge amounts of human and economic resources to develop sophisticated biped robots prototypes [1], [2], [3]. However, some researchers have followed a rather low-cost biped robot design philosophy. Such kind of biped robots is similar to its costly counterpart affording similar capacities to study and improve new biped walking algorithms that in turn have resulted rather convenient. Therefore the trend of building low-cost biped robots has been increasing worldwide [4], [5].

In traditional legged robots, stability is maintained by keeping at least three contact points on the ground surface at all times. Within biped machines, only two points are actually contacting the ground surface that endorses the importance of implementing novel algorithms to achieve balance.

There are some techniques to implement a balance control of a biped robot. Many of them are implemented using classic control techniques while some others use either soft-computing or artificial intelligence techniques. In this work, an incremental fuzzy PD controller is employed to achieve balance on a biped robot [6]. One hybrid dynamic approach model for biped robots is proposed. It combines the inverted pendulum model approach to characterize the biped's walking and one back-propagation

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 141-150

neural network system identification approach to model the biped's balance. The neural network, predicts the behavior of the ZMP during walking.

In order to test the balance control, the "Dany Walker" biped robot structure was designed. It has 10 degrees of freedom (DOF) with each joint being driven by a DC-servo motor (Fig. 1 left). One modular design was chosen to allow an easy assembly and to ease reconfiguration to support several DOF setups. In real-time biped robot structures, a feedback-force system at each foot has to be implemented to obtain the ZMP that is then fed into the incremental fuzzy PD controller, calculating the ZMP error. Then the controller adjusts the lateral robot's positions to keep the ZMP point within the support region [6].

The dynamic of a biped robot is closely related to its structure and its mass distribution [5]. Therefore the movement of the Center of Mass (COM) will have a significant impact on the overall robot stability.

In order to achieve static stability, we place the COM as lower as possible. To such a purpose, a short leg's position was used (Fig. 1, left and right).

To compensate the disturbances during walking, lateral movements of the robot were enabled by mechanical design. Thus, it was possible to control the lateral balance of the robot by swinging the waist using 4 motors, two at the waist and two over the ankles (Fig. 1 right).

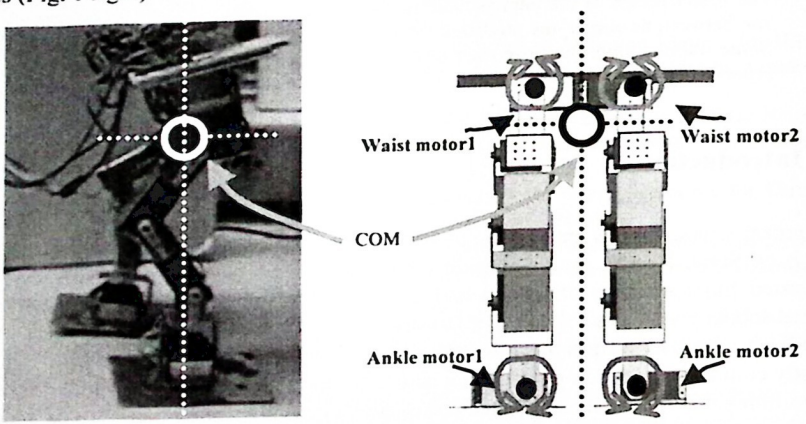


Figure 1. Designed biped robot structure "Dany Walker": left) Real structure, right) CAD design.

This paper is organized as follows: In section 2, the biped balance theory is briefly described. Section 3 presents the dynamic robot's model while Section 4 discusses some conclusions.

2. Biped balance theory

In dynamic walking, the important control criterion is to keep the *Zero Moment Point* (*ZMP*) within the support region (from now on, this criteria is mentioned as the "*ZMP criteria*"). The use of *ZMP criteria* has been broadly used to generate biped control algorithms [2], [3].

2.1. ZMP

The *ZMP* represents a point on the ground where the sum of all momentums is zero. Using this principle, the *ZMP* can be computed as follows:

$$x_{ZMP} = \frac{\sum_i m_i(z+g)x_i - \sum_i m_i \ddot{x} z_i - \sum_i I_{iy}\dot{\theta}_{iy}}{\sum_i m_i(z+g)} \quad (1)$$

$$y_{ZMP} = \frac{\sum_i m_i(z+g)y_i - \sum_i m_i \ddot{y} z_i - \sum_i I_{ix}\dot{\theta}_{ix}}{\sum_i m_i(z+g)} \quad (2)$$

where (x_{ZMP}, y_{ZMP}) are the *ZMP* coordinates, (x_i, y_i, z_i) is the mass centre of the link *i* in the coordinate system, *m* is the mass of the link *i*, and *g* is the gravitational acceleration. I_{ix} and I_{iy} are the inertia moment components, θ_{iy} and θ_{ix} are the angular velocity around the axes *x* and *y* (taken as one point from the mass centre of the link i). The biped balance is achieved when the ZMP is controlled and continuously corrected to fall inside of the boundaries of the support region [6].

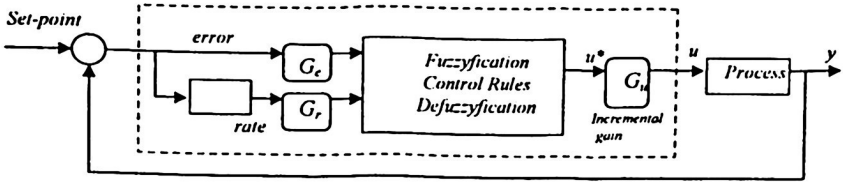


Fig. 2. Fuzzy PD incremental algorithm structure.

2.2. Balance control algorithm

In this work, the fuzzy PD algorithm for incremental control is implemented to balance the robot [6]. The fuzzy PD incremental control algorithm has the structure illustrated in Fig. 2. Gains *Gu*, *Ge* and *Gr* represent the output gains as determined by tuning. They correspond respectively to the error (*ZMP error*) and error rate (*ZMP rate*) gains. The value u^* is the defuzzyficated output, also known as "crisp output". The value *u* is defined by:

Fig. 6 shows the architecture used to train a back-propagation neural network and identify the biped robot's ZMP dynamic model. First, from the real biped robot structure, (real robot's dynamics) the ZMP is obtained (ZMP (k)) and feed it to the incremental fuzzy PD controller. The controller produces an output *M(k)* (lateral motors output) to correct the ZMP inside of the support polygon. *M(k-1)*, *M(k-2)* and *ZMP(k-1)* are respectively the controller output delayed one time unit, two time unit, and the ZMP delayed one time unit.

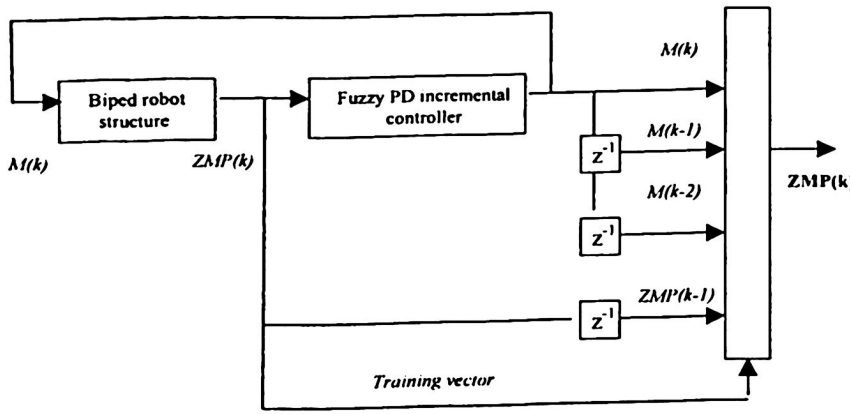


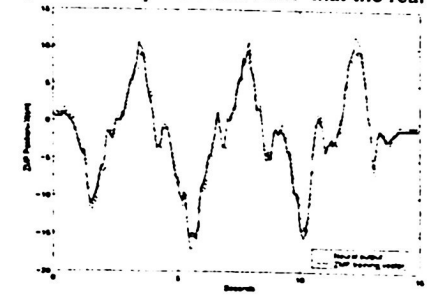
Fig. 6. Architecture to identify the biped robot's ZMP dynamic model.

Thus, to model the biped robot's balance dynamics, a back propagation neural network with four input neurons and an output neuron and with linear output activation function, was choose. The network was trained offline in batch mode, using data collected from the real walking operation of the biped robot. Some different training algorithms were tested for the network training, each; obtain a different biped robot's ZMP dynamic model performance.

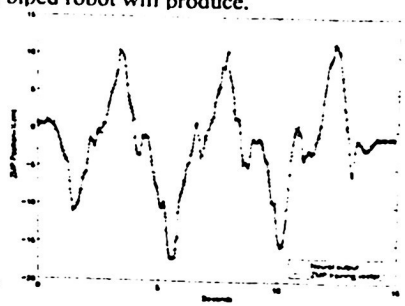
3.1.1. Neural Network model's performance

The in general a neural network performance could depend on many factors, including the complexity of the problem, the number of data points in the training set, the number of weights and biases in the network, the error goal, and the application it self (discriminant analysis, regression, etc). The last, is our case, since the goal is to find, means a neural network, a function approximation which model the biped robot's ZMP dynamic. The criteria to know which training algorithm better describes the ZMP robot's dynamic at walking will be a compromise between the velocity and economy of the algorithm.

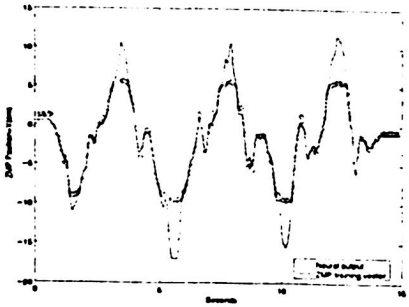
The neural network was training using different training methods. To test the performance of each of them, the controller's output at walking was feed to the neural network. Expecting that the neural network, now trained with the biped's ZMP dynamics, be able to predict the ZMP that the real biped robot will produce.



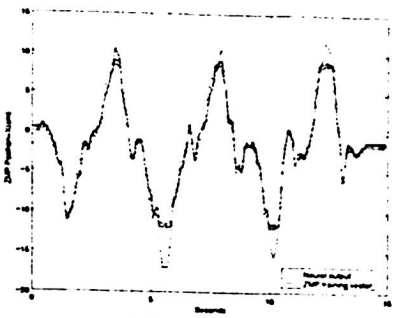
a) Levenberg-Marquardt



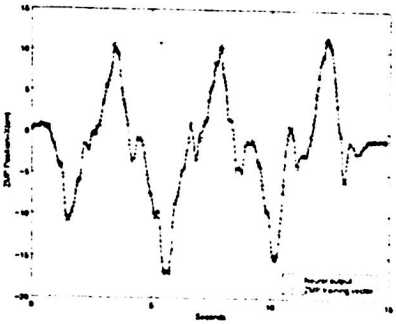
b) Resilient Back-propagation



c) Scaled Conjugate Gradient



d) One-Step Secant



e) BFGS Quasi-Newton

Fig. 7. Performance of some neural network training algorithms to proximate the real ZMP.

In Figure 7 a data set of ZMP values obtained at real walking, are compared with the ZMP produced by the neural network using different training algorithms. Fig. 7, shows the performance with the next training algorithms: a) Levenberg-Marquardt, b) Resilient Back-propagation, c) Scaled Conjugate Gradient, d) One-Step Secant and e) BFGS Quasi-Newton (Broyden, Fletcher, Goldfarb, and Shanno (BFGS)).

4. Conclusions

A neural network used to model the nonlinear biped robot's lateral movements dynamic was implemented. The strategy was to use a neural network as a system identifier; in this case the system to be identified is the biped robot's lateral movement's dynamics. A part of the lateral movements are generated by the fuzzy controller to correcting the ZMP. The ZMP dynamic was the parameter learned by the neural network. Some different training methods were used to compare the performance of the neural network to approximate the real robot's ZMP dynamic at walking. In all the different training algorithms, a back-propagation neural network architecture was choose. From each tested algorithms can be concluded:

a) Levenberg-Marquardt training algorithm

In general, this algorithm has the fastest convergence on function approximation problems. This advantage is especially noticeable if very accurate training is required. In many cases, Levenberg-Marquardt training algorithm is able to obtain lower mean square errors than any of the other algorithms tested. However, as the number of weights in the network increases, the advantage of the Levenberg-Marquardt training algorithm decreases. The performance of the algorithm to approximate the ZMP dynamics was quite gut (Figure 7(a)). A disadvantage, was that the storage requirements of Levenberg-Marquardt training algorithm were larger than the other tested algorithms.

b) Resilient Back-propagation training algorithm

The Resilient Back-propagation training algorithm is the fastest algorithm on discriminant analysis problems. However, in general it does not perform well on function approximation problems. Its performance also degrades as the error goal is reduced. An advantage is that its memory requirements are relatively small in comparison to the other tested algorithms. Figure 7(b), shows the performance of the resilient back propagation training algorithm to model the biped robot's ZMP dynamics.

c) Scaled Conjugate Gradient (SCG) training algorithm

The SCG algorithm demonstrated to be almost as fast as the Levenberg-Marquardt training algorithm on the approximation of the biped balance dynamics. How ever,

Figure 7 c) shows that its performance to model the biped robot's ZMP dynamics was inferior to the two first training algorithms. An important advantage is that the conjugate gradient algorithm has relatively modest memory requirements.

d) One-Step Secant training algorithm (OSS)

The one step secant (OSS) method is an attempt to bridge the gap between the conjugate gradient algorithms and the quasi-Newton (secant) algorithms. This algorithm does not store the complete Hessian matrix; it assumes that at each iteration, the previous Hessian was the identity matrix. This has the additional advantage that the new search direction can be calculated without computing a matrix inverse. However, Figure 7(d) shows that the performance of the OSS training algorithm to model the biped robot's ZMP dynamics was even inferior to the first tree training algorithms. An advantage is that it required less storage and computation per epoch than the BFGS algorithm, but required slightly more storage and computation per epoch than the conjugate gradient algorithms. It can be considered a compromise between full quasi-Newton algorithms and conjugate gradient algorithms.

e) BFGS Quasi-Newton training algorithm

In Figure 7(e) and 7(a) a similar performance between the Quasi-Newton and Levenberg-Marquardt training algorithm can be observed. The Quasi-Newton does not require as much storage as Levenberg-Marquardt training algorithm, but the computation required does increase geometrically with the size of the network, since the equivalent of a matrix inverse must be computed at each iteration.

As a result from these test, the BFGS Quasi-Newton training algorithm to model the robot's ZMP dynamics is prefer for its convenient relationship between computational economy and fast convergence.

References

[1] Hirai, K.H., Haikawa, M., Takenaka, Y. (1998). The development of Honda humanoid robot. Proceedings. IEEE International Conference in Robotics and Automation. Leuven, Belgium. (1998)

[2] Takanishi, A., Ishida, M., Yamazaki, A., Kato, I. (1985). The realization of dynamic walking robot WL-10RD, in Proc. 1985 Int. Conf. Advanced Robotics, pp. 459-466.

[3] Takanishi, A., Egusa, Y., Tochizawa, M., Takeya, T., Kato, I. (1988). Realization of dynamic biped walking stabilized with waist motion. Proceeding of the Seventh CISMIFTOMM Symposium on Theory and Practice of Robots and Manipulators, pp 68-79.

[4] Nicholls, E.: Bipedal dynamic walking in robotics (1998). Dissertation thesis, University of Western Australia, Department of electrical and Electronics Engineering.

[5] Cuevas, E. V., Zaldivar, D., Rojas, R. (2004). Bipedal robot description, Technical Report B-03-19, Freie Universität Berlin, Fachbereich Mathematik und Informatik, Berlin, Germany.

[6] Cuevas, E. V., Zaldivar, D., Rojas, R. (2005). Incremental fuzzy control for a biped robot balance IASTED International Conference on ROBOTICS AND APPLICATIONS -RA2005- Cambridge, USA.

[7] Norgaard, M., Ravn, O., Poulsen, N. K., Hansen, L. K. (1999). Neural Networks for Modelling and Control of Dynamic Systems. Springer Verlag, Berlin.

RNN Identification and Control of Nonlinear Plants Using the Recursive Marquardt Algorithm

Ieroham Baruch, Saul Escalante M., and Carlos R. Mariaca-Gaspar
CINVESTAV-IPN, Ave. IPN No 2508,
A.P. 14-470 Mexico D.F., C.P. 07360 MEXICO
E-mail: {baruch, sescalante, cmariaca}@ctrl.cinvestav.mx
(Paper received on August 19, 2006, accepted on September 27, 2006)

Abstract. The paper proposed a new Recurrent Neural Network (RNN) model for systems identification and states estimation of nonlinear mechanical plants. The RNN model is learned by the second order recursive learning algoritm of Levenberg-Marquardt (L-M). The estimated states of the recurrent neural network model are used for direct adaptive trajectory tracking control systems design. The system contains also a noise rejection output filter. The applicability of the proposed neural control system is confirmed by simulation results with MIMO mechanical plant and compared with the results obtained by the Backpropagation learning algorithm. The results show a good convergence of both algorithms with priority of the L-M algorithm.

1. Introduction

The recent advances in understanding of the working principles of artificial neural networks (ANN) and the rapid growth of available computational resources led to the development of a wide number of ANN-based modeling, identification, prediction and control applications, [1]-[6], especially in the field of mechanical engineering and robotics. The ability of ANN to approximate complex non-linear relationships without prior knowledge of the model structure makes them a very attractive alternative to classical modeling and control techniques. Many of the applications currently reported are based on the classical Nonlinear Autoregressive Moving Average (NARMA) model, where a Feedforward Neural Network (FFNN) is implemented, [4], [5]. [6]. However, the FFNN has in general a static structure, therefore it is adequate to approximate mainly static (nonlinear) relationships and their real-time applications for dynamical systems require the introduction of external time-delayed feedbacks. [5]. The application of the FFNN for modeling, identification and control of nonlinear dynamic plants caused some problems which could be summarized as follows: 1. The dynamic systems modeling usually is based on the NARMA model which need some information of input/output model orders, and input and output tap-delays ought to be used, [5], [6]; 2. The FFNN application for Multi-Input Multi-Output (MIMO) systems identification needs some relative order structural information; 3. The ANN model structure ought to correspond to the structure of the identified plant where four different input/output plant models are used. [5]; 4. The lack of universality in ANN architectures caused some difficulties in its learning and a Backpropagation through time

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 151-160

learning algorithm needs to be used. [7]; 5. Most of NARMA-based ANN models are sequential in nature and introduced a relative plant-dependent time-delay; 6. Most of the ANN-based models are nonparametric ones. [5], and so, not applicable for an indirect adaptive control systems design; 7. All this ANN-based models does not perform state and parameter estimation in the same time, [4]; 8. All this models are appropriate only for identification of nonlinear plants with smooth, single, odd, nonsingular nonlinearities.

Recurrent Neural Networks (RNN) possesses its internal time-delayed feedbacks, so they are promising alternative for system identification and control, particularly when the task is to model dynamical systems [2], [3], [4], [7]. Their main advantage is the reduced complexity of the network structure. However, the analysis of state of the art in the area of classical RNN-based modeling and control has also shown some of their inherent limitations as follows: 1. The RNN input vector consists of a number of past system inputs and outputs and there is not a systematic way to define the optimal number of past values [4] and usually, the method of trials and errors is performed; 2. The RNN model is naturally formulated as a discrete model with fixed sampling period, therefore, if the sampling period is changed, the network has to be trained again; 3. It is assumed that the plant order is known, which represents a quite strong modeling assumption in general, [5]. Driven by these limitations, a new Recurrent Trainable Neural Network (RTNN) topology and the recursive Backpropagation (BP) type learning algorithm in vector-matrix form was derived, [8], [9], [10], and its convergence was studied, [9]. But the recursive BP algorithm, applied for RTNN learning, is a gradient descent first order learning algorithm which not permits to augment the precision and to accelerate the learning. So, the aim of the paper is to apply for RTNN learning a second order algorithm like the Levenberg-Marquardt (L-M) algorithm, [11], [12], [13], it is. The RTNN and the L-M algorithm of its learning will be applied for identification and control of a mechanical MIMO plant, taken from [6].

2. Topology and Learning of the RTNN

RTNN Topology: A *Recurrent Trainable Neural Network* model and its learning algorithm of dynamic *Backpropagation-type*, together with the explanatory figures and stability proofs, are described in [9]. The RTNN topology, given in vector-matrix form is described by the following equations:

$$X(k+1) = AX(k) + BU(K) \tag{1}$$

$$Z(k) = S[X(k)] \tag{2}$$

$$Y(k) = S[CZ(k)] \tag{3}$$

$$A = block - diag(A_{ii}); |A_{ii}| < 1 \tag{4}$$

Where: Y, X, and U are, respectively, output, state and input vectors with dimensions l, n, m; A is a (nxn)- state block-diagonal weight matrix; A_{ii} is an i-th diagonal block of A with (lxl) dimension. Equation (4) represents the local stability conditions, imposed on all blocks of A; B and C are (nxm) and (lxn)- input and output weight matrices; S is

vector-valued sigmoid or hyperbolic tangent-activation function, [9]; k is a discrete-time variable. The stability of the RTNN model is assured by the activation functions and by the local stability condition (4).

BP Learning of the RTNN: The commonly used BP updating rule, [9], is given by:

$$W_{ij}(k+1) = W_{ij}(k) + \eta \Delta W_{ij}(k) + \alpha \Delta W_{ij}(k-1) \tag{5}$$

Where: W_{ij} is a general weight, denoting each weight matrix element (C_{ij}, A_{ij}, B_{ij}) in the RTNN model, to be updated; ΔW_{ij} (ΔC_{ij}, ΔJ_{ij}, ΔB_{ij}), is the weight correction of W_{ij}; while; η and α are learning rate parameters. The weight updates are computed by the following equations:

$$\Delta C_{ij}(k) = [T_i(k) - Y_i(k)] S_i'[Y_i(k)] Z_j(k) \tag{6}$$

$$\Delta A_{ij}(k) = R X_j(k-1) \tag{7}$$

$$R = C_i(k)[T(k) - Y(k)] S_j'[Z_i(k)] \tag{8}$$

$$\Delta B_{ij}(k) = R U_j(k) \tag{9}$$

Where: ΔA_{ij}, ΔB_{ij}, ΔC_{ij} are weight corrections of the weights A_{ij}, B_{ij}, C_{ij}, respectively; $(T\text{-}Y)$ is an error vector of the output RTNN layer, where T is a desired target vector and Y is a RTNN output vector, both with dimensions l; X_i is an i-th element of the state vector; R is an auxiliary variable; S_j' is derivative of the activation function. Stability proof of this learning algorithm is given in [9]. The described above RTNN is applied for identification and adaptive control of a nonlinear MIMO plant.

Recursive Levenberg-Marquardt Algorithm for RTNN Learning: The recursive L-M algorithm of learning, [11], [12], [13], is given by the following equations:

$$W(k+1) = W(k) + P(k) \nabla Y[W(k)] e[W(k)] \tag{10}$$

$$Y[W(k)] = g[W(k), U(k)] \tag{11}$$

$$E[W(k)] = e^2[W(k)] = \{g[W(k), U(k)] - Y_p(k)\}^2 \tag{12}$$

$$\nabla Y[W(k)] = \frac{\partial}{\partial W} g[W, U(k)] \Big|_{W = W(k)} \tag{13}$$

The Jacobean matrix elements for the RTNN topology are as follows:

$$\nabla Y_C[C_{ij}(k)] = S_j'(Y_j(k)) Z_i(k) \tag{14}$$

$$\nabla Y_A[A_{ij}(k)] = R_i X_j(k-1) \tag{15}$$

$$\nabla Y_B[B_{ij}(k)] = R_i U_j(k) \tag{16}$$

$$R_i = C_i(k) S_j'(Z_i(k)) \tag{17}$$

So the Jacobean matrix could be formed as:

$$\nabla Y[W(k)] = [\nabla Y_C(C_{ij}(k)), \nabla Y_A(A_{ij}(k)), \nabla Y_B(B_{ij}(k))] \tag{18}$$

The P(k), S(.), and Ω(.) matrices are given as follows:

$$P(k) = \frac{1}{\alpha(k)}\{P(k-1) - P(k-1)\Omega[W(k)]S^{-1}[W(k)]\Omega^T[W(k)]P(k-1)\} \tag{19}$$

$$S[W(k)] = \alpha(k)\Lambda(k) + \Omega^T[W(k)]P(k-1)\Omega[W(k)] \tag{20}$$

$$\Omega^T[W(k)] = \begin{bmatrix} & \nabla Y^T[W(k)] & \\ 0 \quad \cdots & 1 & \cdots \quad 0 \end{bmatrix}; \Lambda(k)^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & \rho \end{bmatrix}; \tag{21}$$

$$0.97 \le \alpha(k) \le 1$$

The next part compares the simulation results obtained with both BP and L-M learning algorithms for MIMO nonlinear plant.

3. Direct Adaptive Neural Control System Structure

The block-diagram of the control system is given on Figure 2. It contains a recurrent neural identifier RTNN-1, two neural controllers (feedback and feedforward) RTNN-2, RTNN-3, and a low pass noise rejection filter. In the direct adaptive neural control, the weight parameters of the feedback and feedforward controllers are learned so to minimize the cost function which is the reference tracking quadratic instantaneous error of the plant output.

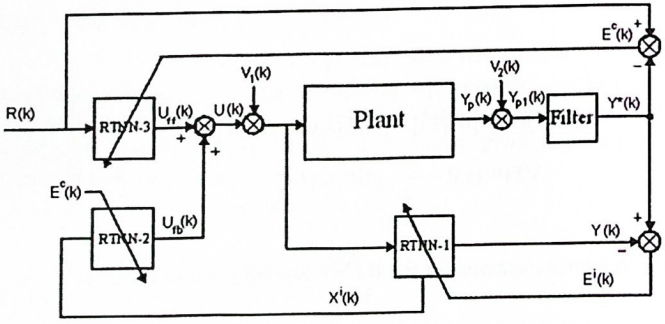


Fig. 1. Block - diagram containing neural identifier, feedback and feedforward neural controllers, and plant output noise rejection filter.

The structure of the closed loop system contains a neural identifier issuing an estimated state vector to the feedback neural plant dynamics compensator. The control feedback

signal is added to the signal of the feedforward neural controller. The feedforward controller represented in fact an inverse model of the feedback closed loop system and depends on the reference signal. The system is completed by a low-pass filter which is aimed to reject the plant and measurement noises. So, the RTNN-1 identified the combined dynamics of the plant and the filter, and estimates the stated of this complex dynamics. The plant and the filter (this also includes the input and output noises) are described by the following state-space nonlinear models:

$$X_p(k+1) = F[X_p(k), U(K), V_1(k)] \tag{22}$$

$$Y_p(k) = \varphi(X_p(k)) \tag{23}$$

$$Y_{p1}(k) = Y_p(k) + V_2 \tag{24}$$

$$Y^*(k+1) = A^*Y^*(k) + B^*Y_{p1}(k) = A^*Y^*(k) + B^*Y_p(k) + B^*V_2 \tag{25}$$

Here the input, output and state dimensions of the plant are m, l, n_p. The filter dynamics is completely decupled, so the state matrix is diagonal (lxl) one. The plant equations (22) and (23) could be linearized and written in the same state-space form as:

$$X_p(k+1) = A_pX_p(k) + B_pU(k) + B_pV_1(k) \tag{26}$$

$$Y_p(k) = C_pX_p(k) \tag{27}$$

The linearized identification RTNN-1 could be also described by a state space model:

$$X'(k+1) = A'X'(k) + B'U(k) \tag{28}$$

$$Y'(k) = C'X'(k) \tag{29}$$

Here the input, output and state dimensions of the RTNN-1 are m, l, n_t. The feedback neural RTNN-2 controller has a similar linearized state space representation, which input is the estimated systems state, issued by the RTNN-1:

$$X^c{}_{fb}(k+1) = A^c{}_{fb}X^c{}_{fb} + B^c{}_{fb}X'(k) \tag{30}$$

$$U_{fb}(k) = -C^c{}_{fb}X^c{}_{fb}(k) \tag{31}$$

Here the input, output and state dimensions of the RTNN-2 are n_t, m, n_{fb}. The feedforward neural RTNN-3 controller could be described in the same manner as:

$$X^c{}_{ff}(k+1) = A^c{}_{ff}X^c{}_{ff} + B^c{}_{ff}R(k) \tag{32}$$

$$U_{ff}(k) = C^c{}_{ff}X^c{}_{ff}(k) \tag{33}$$

Here the input, output and state dimensions of the RTNN-3 are l, m, n_{ff}.

Let us to write the following z-transfer functional representations of the given up state-space equations for the plant, filter, feedback and feedforward controllers:

$$W^{P}(z)=C^{P}\left(zI-A_{P}\right)^{-1}B_{P};W^{*}(k)=C^{*}\left(zI-A^{*}\right)^{-1}B^{*};P^{I}(z)=\left(zI-A^{I}\right)^{-1}B^{I} \tag{34}$$

$$Q_{1}(z)=C^{I}{}_{fb}\left(zI-A^{I}{}_{fb}\right)^{-1}B_{fb};Q_{2}(z)=C^{I}{}_{ff}\left(zI-A^{I}{}_{ff}\right)^{-1}B_{ff} \tag{35}$$

The control systems z-transfer functions (34), (35) are connected by the following equations, given in z-operational form:

$$Y^{*}(z)=W^{*}(z)Y_{p}+W^{*}(z)V_{2};X^{I}(z)=P^{I}(z)U(z) \tag{36}$$

$$U_{fb}(z)=-Q_{1}(z)X^{I}(z);U_{ff}(z)=Q_{2}(z)R(z) \tag{37}$$

$$Y_{p}(z)=W_{p}(z)U(z)+W_{p}V_{1}(z);U(z)=U_{ff}(z)+U_{fb}(z) \tag{38}$$

Effectuating some substitutions and mathematical manipulations we could obtain the following statement for the systems control variable:

$$U(z)=\left[I+Q_{1}(z)P^{I}(z)\right]^{-1}Q_{2}(z)R(z) \tag{39}$$

The substitution of the control into the plant equation yields:

$$Y_{p}(z)=W_{p}(z)\left[I+Q_{1}(z)P^{I}(z)\right]^{-1}Q_{2}(z)R(z)+W_{p}(z)V_{1}(z) \tag{40}$$

The substitution of the plant equation into the systems output equation finally gives:

$$Y^{*}(z)=W^{*}(z)W_{p}(z)[I+Q_{1}(z)P^{I}(z)]^{-1}Q_{2}(z)R(z)+V_{3}(z) \tag{41}$$

Where $V_3(.)$ is a generalized noise term, given as:

$$V_{3}(z)=W^{*}(z)[W_{p}(z)V_{1}(z)+V_{2}(z)] \tag{42}$$

The RTNN topology is controllable and observable, [9], and the L-M algorithm of learning are convergent, so the identification and control errors tend to zero ($E^{I}(k)=Y^{*}(k)-Y^{I}(k)\to 0;\quad k\to\infty;E^{c}(k)=R(k)-Y^{*}(k)\to 0;\quad k\to\infty$) which means that each transfer functions given by equations (34), (35) is stable with minimum phase. From (41) it is seen that the dynamics of the stable low pass filter is independent from the dynamics of the plant and it does not affects the stability of the closed-loop system. The closed-loop system is stable and the RTNN-2 controller compensates the combined "plant plus filter" dynamics. The RTNN-3 feedforward controller dynamics is an inverse dynamics of the closed-loop systems one, which assure a precise reference tracking in spite of the presence of process and measurement noises.

4. Simulation Results

Let us consider the MIMO mechanical plant governed by the following state-space discrete-time nonlinear dynamics equations, taken from [6]:

$$X_1(k+1) = 0.9X_1(k)\sin[X_2(k)] + \left[2 + 1.5\frac{X_1(k)U_1(k)}{1+X^2_1U^2_1(k)}\right]U_1 + \left[X_1(k) + \frac{2X_1(k)}{1+X^2_1(k)}\right]U_2 \quad (43)$$

$$X_2(k+1) = X_1(k)\{1+\sin[2X_1(k)]\} + \frac{X_1(k)}{1+X^2_1(k)} \quad (44)$$

$$X_3(k+1) = \{3+\sin[2X_1(k)]\}U_2(k) \quad (45)$$

$$Y_1(k) = X_1(k); Y_2(k) = X_2(k) \quad (46)$$

The input, output and state dimensions of the plant are 2, 2, and 3. The reference signals R_1, R_2 of the control system are chosen as:

$$R_1(k) = 0.007\sin(\pi k/10) + 0.0075\sin(\pi k/25) + 0.006\sin(\pi k/50) \quad (47)$$

$$R_2(k) = 0.0075\sin(\pi k/40) + 0.007\sin(\pi k/60) + 0.006\sin(\pi k/80) \quad (48)$$

The control systems structure was applied to the given up plant and the results of its functioning are given on Fig. 2, Fig.3, Fig. 4 for BP and L-M learning applied for systems with 10% process and measurement noises, with and without noise filter. The results obtained for control MSE% are summarized in Table 1.

Table 1. The MSE% of control given for BP and L-M algorithms of learning for control systems with and without noise filter

Learning / noise filtering	MSE1 (%)	MSE2 (%)
BP with filter	0.91	0.72
BP without filter	1.28	2.19
L-M with filter	0.66	0.64
L-M without filter	1.03	1.41

The results show the good convergence of both learning algorithms. The noise terms augmented the MSE% of control in both cases for systems without noise filters. The L-M algorithm of learning is more precise but more complex that the BP one.

5. Conclusions

The paper proposed a new RTNN model for systems identification and states estimation of nonlinear mechanical plants. The RTNN is learned by the second order recursive learning algorithm of Levenberg-Marquardt. The estimated states of the recurrent

neural network model are used for direct adaptive trajectory tracking control systems design. The system contains also a noise rejection output filter, which dynamics is separated from the dynamics of the control system. The applicability of the proposed neural control system is confirmed by simulation results with a MIMO mechanical plant and compared with the results obtained by the BP learning algorithm. The results summarized in Table 1 show good convergence of both L-M and BP learning algorithms. The presence of noise terms augmented the MSE% of control in both cases of systems without noise filters. The L-M algorithm of learning is more precise but more complex then the BP one.

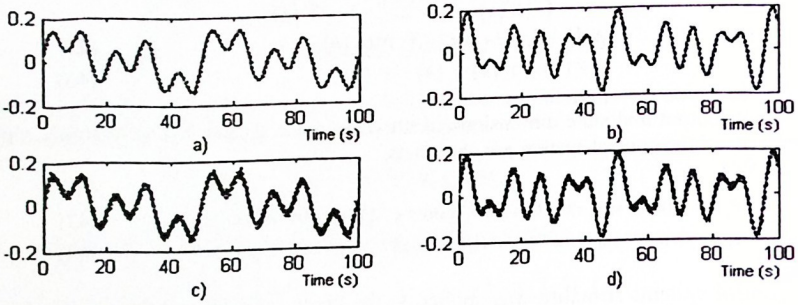


Fig. 2. Comparative results of a direct adaptive trajectory tracking control with BP learning, with (a,b) and without (c,d) noise filtering; a) Comparison between the first plant output Yp1(doted line, d-l) and the reference signal R1 (continuous line, c-l) for the case of system with noise filtering; b) Comparison between the second plant output Yp2 (d-l) and the reference signal R2 (c-l) for the case of system with noise filtering; c) Comparison between the first plant output Yp1(d-l) and the reference signal R1 (c-l) for the case of system without noise filtering; d) Comparison between the second plant output Yp2 (d-l) and the reference signal R2 (c-l) for the case of system without noise filtering.

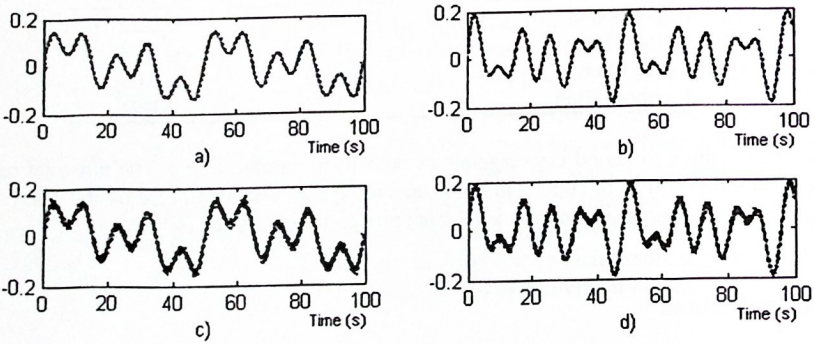


Fig. 3. Comparative results of a direct adaptive trajectory tracking control with L-M learning with (a,b) and without (c,d) noise filtering; a) Comparison between the first plant output Yp1(d-l) and the reference signal R1(c-l) for the case of system with noise filtering; b) Comparison

between the second plant output Yp2(d-l) and the reference signal R2(c-l) for the case of system with noise filtering; c) Comparison between the first plant output Yp1(d-l) and the reference signal R1(c-l) for the case of system without noise filtering; d) Comparison between the second plant output Yp2(d-l) and the reference signal R2(c-l) for the case of system without noise filtering.

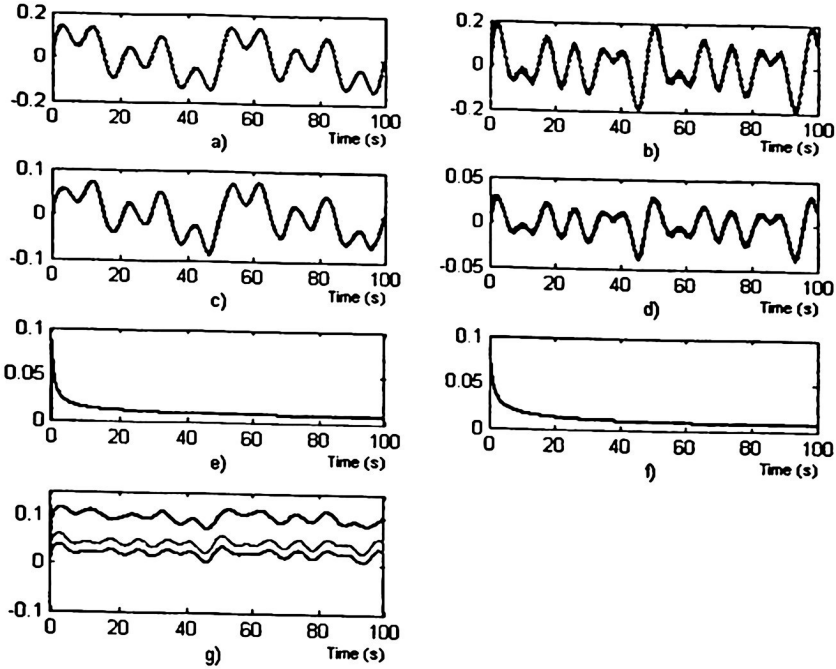


Fig. 4. Some additional results of a direct adaptive trajectory tracking control with L-M learning with noise filtering; a) Comparison between the first plant output Yp1 (c-l) and the first output Yi1 (d-l) of the identification RTNN-1; b) Comparison between the second plant output Yp2 (c-l) and the second output Yi2(d-l) of the identification RTNN-1; c) First control signal U1; d) Second control signal U2; e) Mean squared error of control (MSE1%) for the first controlled output; f) Mean squared error of control (MSE2%) for the second controlled output; g) Systems state variables, estimated by RTNN-1.

Acknowledgements

We want to thank CONACYT-MEXICO for the scholarship given to Saul Fernando Escalante Magana, MS student and Carlos Roman Mariaca Gaspar, PhD student, both students at the CINVESTAV-IPN, MEXICO.

References

[1] Miller III, W.T., Sutton, R.S., Werbos P.J. (1992). Neural Networks for Control, MIT Press, London.

[2] Chen, S., Billings, S. A. (1992). Neural Networks for Nonlinear Dynamics System Modeling and Identification. International Journal of Control, **56:**319-346.

[3] Pao, S.A., Phillips, S.M., Sobajic, D. J. (1992). Neural Net Computing and Intelligent Control Systems. International Journal of Control, **56:**263-289.

[4] Hunt, K.J., Sbarbaro, D., Zbikowski, R., Gawthrop, P.J. (1992). Neural Network for Control Systems - A Survey. Automatica, **28** (6):1083-1112.

[5] Narendra, K.S., Parthasarathy, K. (1990). Identification and Control of Dynamic Systems Using Neural Networks. IEEE Transactions on Neural Networks, 1(1):4-27.

[6] Narendra K., Mukhopadhyay S. (1994). Adaptive Control of Nonlinear Multivariable Systems using Neural Networks. Neural Networks, 737-752.

[7] L. Jin, and M. Gupta (1999). Stable Dynamic Backpropagation Learning in Recurrent Neural Networks. IEEE Transactions on Neural Networks, **10:**1321-1334.

[8] Baruch I., J.M., Flores, F., Thomas and R. Garrido. (2001). Adaptive Neural Control of Nonlinear Systems. In: Dorffner, G., Bischof, H., Hornik, K. (eds.): Artificial Neural Networks-ICANN 2001, Lecture Notes in Computer Science Vol. 2130. Springer, Berlin, pp. 930-936.

[9] Baruch, I. S., Flores, J. M., Nava, F., Ramirez, I. R., Nenkova, B. (2002). An Adavanced Neural Network Topology and Learning, Applied for Identification and Control of a D.C. Motor. In: Sgurev, V.S., Jotsov, V. (eds.): Proc. of the First Int. IEEE Symposium on Intelligent Systems, Varna, Bulgaria, pp. 289-295.

[10] Flores, J. M., Baruch, I., Garrido, R. (2001). Red Neuronal Recurrente para Identificación y Control de Sistemas No Lineales. Cientifica-ESIME-IPN, **5** (1):11-20.

[11] Asirvadam, V. S., McLoone, S. F. and Irwin G. W. (2002). Parallel and Separable Recursive Levenberg-Marquardt Training Algorithm. In: Proceedings of the 2002 12th IEEE Workshop on Neural Networks for Signal Processing, pp. 129-138.

[12] Ngia, L. S., Sjöberg J., and Viberg, M. (1998). Adaptive Neural Nets Filter Using a Recursive Levenberg Marquardt Search Direction. IEEE Signals, Systems and Computer, **1:** 697-701.

[13] Ngia, L. S., Sjöberg, J. (2000). Efficient Training of Neural Nets for Nonlinear Adaptive Filtering Using a Recursive Levenberg Marquardt Algoritm. IEEE Trans. on Signal Processing, **48:**1915-1927.

Forward Models and the Prediction of Undesired Situations

Bruno Lara, Juan M. Rendon, and Marcos Capistrán

Facultad de Ciencias
Universidad Autónoma del Estado de Morelos
Av. Universidad 1001, Col. Chamilpa
Cuernavaca, Mexico
{bruno.lara, rendon, marcos}@uaem.mx

(Paper received on August 10, 2006, accepted on September 25, 2006)

Abstract. The research presented in this paper highlights the importance of prediction and action as part of the perceptual process of a cognitive system. Through interaction with its environment a forward model is trained to predict multi-modal sensory representations formed by visual and tactile stimuli. The forward model is mounted on a simulated artificial agent and implemented as an artificial neural network, the data used for training comes from the simulated environment of the agent. Once trained, the predictions of the network are thoroughly analysed, the forward model is then implemented on the simulated agent in a typical obstacle avoidance task with a phototropic behaviour. Results show that forward models can represent a very important tool for the behaviour of autonomous agents.

1 Introduction

Classical views of cognition explain behaviour as a product of a direct, unidirectional line of information processing. Sensory inputs create a sensory representation and according to this a motor action is performed, actions are regarded as reactions, responses to stimuli. Most of the observed behaviour is considered a consequence of an innate stimulus-response mechanism that is available to the individual [1]. Known as the *information processing metaphor*, this framework thinks of the perception processes as modules that receive, modify and then pass the information available from the environment.

A novel approach to perception considers sensory input and action (motor output) as part of the same cognitive process. Only in recent times the idea that the anticipation of actions and/or sensory states influence behaviour is been appreciated. In the field of cognitive psychology these ideas have recently received much attention. Anticipations are now seen to play an important role on the coordination, planning and realisation of behaviour [2]. The linear information processing approach has given way to new frameworks according to which the direction of information flow is not anymore a one-way path.

At the centre of this view is the importance played by the body of the agent and its dynamic relation with the environment. An interesting proposal is presented by Hommel et al. [3]: The Theory of Event Coding (TEC) which is "... *based on the*

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 161-170

central notion that perception, attention, intention and action share, or operate on, a common representational domain"[3]. TEC links perception and action functionally. It is this link and its coordination that provides the basis for adaptive behaviour [3].

Within this framework, sensory representations are also considered as consequences of actions. Any action realised by an agent on the environment has effects (action effects) and are the main reasons for behaviour. Representations that code for the environmental and bodily consequences of a movement become associated to motor representations coding for that actual movement [4]. The planning and control of actions becomes anticipatory when it is driven by the desired sensory situations or desired action effects.

The cornerstone of this research is the importance of predictions and actions as part of the perceptual process of the cognitive system. The cognitive model is learned and tested through interaction of the agent with its surroundings. The qualitative testing of the decisions the agent takes is based on the needs of the agent to act within its own world. For these reasons, we argue that the agent is grounded in its environment [5].

The objective of the research reported here is the better understanding of the effects and extent of the abilities the implementation of cognitive models affords an artificial agent.

2 Forward Models

A computational equivalent proposal are forward models. Mainly used in the field of motor control, a forward model is an internal model which incorporates knowledge about sensory changes produced by self-generated actions of an agent ([6],[7]). Given a sensory situation S_t and a motor command M_t(intended or actual action) the forward model predicts the next sensory situation S_{t+1}.

Forward models provide an alternative to the classical information processing views on perception. Möller [8] suggested forward models as a possibility to integrate visual perception and action generation.

In the realm of artificial autonomous agents anticipation and forward models can be used as a base for coherent behaviour. Autonomous agents interact with their environment in a direct way. A basic need for them to deal with their world is to predict the events happening. An anticipatory agent learning and using a forward model should be able then to have sufficient information to form planning strategies avoiding undesired situations and reacting timely to the hazards of its environment. Very interesting results have been presented by Dearden et al. [9], where a robot learns a forward model that successfully imitates actions presented to its visual system. Other implementations of cognitive approaches are discussed in Section 5.

The work presented in this paper attempts to provide an artificial agent with the necessary tools to predict undesired situations. The prediction of the agent is based on the inputs to the forward model, this prediction is characterized by the association of visual and tactile stimuli. In the context of TEC, this association can be considered an event composed by the motor command and the sensory situations (actual and desired) [3].

3 Experiments

The forward model is obtained by trainning an artificial neural network (ANN) with data coming from a simulated agent, the network is then tested on trajectories not seen during training. The whole system is implemented on the artificial agent to solve an obstacle avoiding task while seeking for a light source.

3.1 Environment and Data Collection

Using a robot simulator (Fig. 1a) a robot is placed in an arena with obstacles varying in size from 25-50 pixels. In the Figure the robot is moving forward from left to right.

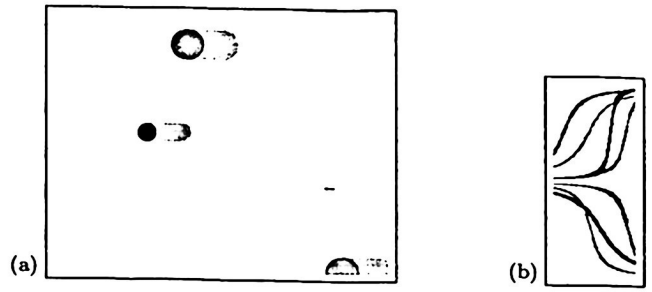


Fig. 1. Virtual World of the robot (a) and the recording of a single trajectory (b).

The robot has a diameter of 30 pixels and is equipped with an omni-directional linear black and white camera and a simulated frontal bumper. The robot moves in a straight line through the arena, taking snapshots of the environment. Every 20 pixels a new image is recorded. Figure 1b shows an 80 steps or 1600 pixels trajectory.

In Figure 1b the y axis represents the spatial dimension of the image, this is, the 360 degrees of the robot view and the x axis represents the time dimension. The front of the robot is located in the middle of the image in the spatial dimension. In the first snapshot, the obstacles can be seen close together and of a relatively small size. As the robot moves forward, the obstacles grow and move away from the centre of the robot until they are at the back of it, (ie. in the far right and far left of the image).

Obstacles are randomly placed to the right, left and front of the robot trajectory. The task of the robot is to predict whether it can perform a collision free trajectory of 1600 pixels. Figure 2 shows two trajectories where a collision occurs forcing it to stop (for convinience the trajectories are rotated 90 degrees). As the robot aproaches obstacles these grow in the image view, the obstaces are located in the central area of the image as this area represents the front of the robot.

3.2 Data Preprocessing and Forward Model Implementation

Before being used in the system the visual information is preprocessed. Originally, the images coming from the robot camera have a size of 1000 x 1 pixels, individual

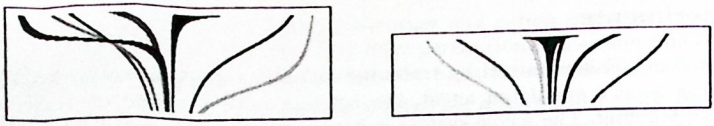


Fig. 2. Trajectory of robot with a crash at the left. Trajectory length: 62 steps. And trajectory of robot with a crash at the front. Trajectory length: 50 steps .

trajectories have different lengths as once the robot encounters an obstacle it stops recording images. The trajectories are preprocessed as follows:

- A section equivalent to 90 degrees (250 pixels) at the front of the robot is extracted from the whole image. As the system requires to predict collision on the front of the robot, this is the visual information relevant to the task.
- A butterworth one dimensional low-pass filter is applied to the spatial dimension of the images. This is done in order to get rid of high frequency redundant information.
- Foveal mapping in the spatial domain. Foveal mapping is basically a weighted subsampling of the image. The farther pixels are from the centre of the image the more they are subsampled, the pixels at the centre remaining nearly unchanged. For the subsampling an averaging mask was used.

The effects of applying the preprocessing algorithms to the image shown in Figure 1b can be seen in Figure 3.



Fig. 3. Frontal 90° after filtering (a) and after fovealisation (b).

A system is needed capable of predicting visual information as well as the simulated bumper states. This system can be implemented as a forward model of the form seen in Figure 4, where the current sensory situation is composed by the visual images V at times t, $t+1$ and $t+2$. This form of the input is expected to provide the model with the necessary information to learn the temporal structure of the data. The output of the forward model is the visual scene and the bumper state for time $t+3$, this is, V_{t+3} and B_{t+3} respectively.

The system performs a local symmetrical prediction. This is, every predictor takes its input from a section of the sensory input and predicts only the central pixel of that section. On the edges of the image this is not possible, so the input window is shifted in order to use the available information.

Given that the final size of the images is 50 pixels, the system consists of 50 feed-forward, fully connected, multi-layer perceptron networks, using sigmoid activation functions. The networks are trained using Resilient Back-propagation (RPROP) [10].

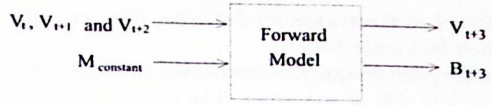


Fig. 4. Particular Forward Model

RPROP is a method to train MLP networks with a weight specific adaptation rule depending on the current and past values of each weight partial derivatives. This adaptive method has the effect of accelerating the learning process.

Each network has the following structure:

- Intput layer: 45 input units that code the visual data of three time steps (15 pixels for each time step).
- Hidden layer: 10 units, which at the time were a compromise between performance, network complexity and training time. Currently we are working on a *EEG* type of analysis of the dynamics of the hidden units activation. With this analysis, we hope to learn among other things whether the size of the hidden layer is the optimal one for the task.
- Output layer: 2 units, one corresponding to the predicted visual value and one for the predicted bumper value (0 for no collision and 1 for collision).

Training patterns for each network are prepared consisting of several images (46000 patterns) with a mixture of different collisions and collision-free trajectories. It is important to note that each one of the networks gets the same bumper value assigned, so that when there is a collision *all* networks should have an activation value of 1.

3.3 Testing the Forward Model

After training the networks for 6000 cycles of batch training, the testing is done with trajectories that were unseen by the network during training. The networks are expected to perform two kinds of prediction. First a one step prediction (OSP), this is, given the values of V_t, V_{t+1} and V_{t+2} the values of V_{t+3} and B_{t+3} are predicted. This is the standard network output. Second a *long term prediction (LTP)* which consists of using the predicted visual data back as input to the system. This prediction compares to an internal simulation of the events.

The OSP bumper values are not binary, instead the activation increases as the robot approaches an obstacle. More importantly, the networks that increase activity are those on the side of the robot where the obstacle is approaching or where significant changes on the visual field occur. A threshold is implemented during OSP to indicate a collision, if 5 or more neurons show an activation higher than 0.5 it means there is a collision.

The system should have a necessity to trigger LTP and this is also defined as a threshold. When 3 or more bumper output neurons present an activation greater than 0.3 during OSP, an internal simulation of the rest of the trajectory starts. Although the threshold might seem low it guarantees that there is no false predictions. The system was tested on 30 different trajectories. It never failed to trigger LTP when there was in

reality a (future) collision on the trajectory. Likewise, it never triggered the LTP when presented with collision-free trajectories.

The visual output a typical testing run can be seen in Fig 5 which shows a trajectory with a collision on the left side of the robot. The system triggers LTP 2 steps before it actually happens, in the robot *world* this means 40 pixels or more than once its own size of 30 pixels.



Fig. 5. System evaluation during a collision on the left side of the robot

Similar behaviours are observed during different trajectories. The system triggers LTP when a collision is likely to occur, most importantly during LPT the activation of the bumper units signals the presence of a crash.

The activation of the whole array of bumper output neurons for the last step of the LTP for three different collisions is presented in Fig. 6. It can be seen that the output neurons presenting higher activation are those located on the side of the robot where the collision is going to occur. In the case of the frontal collision, the activation is higher on the networks where changes in the visual field occur.

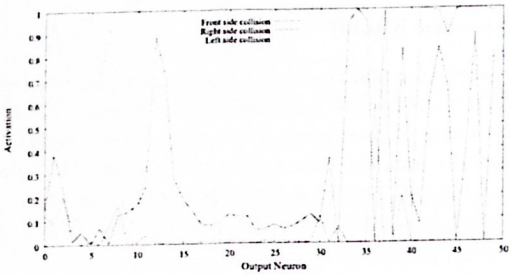


Fig. 6. Activation of the output neurons for bumper states at the last step of LTP

It is worth noting that Fig. 2 show trajectories and collisions before pre-processing. Fig 5 shows a trajectory with a collision after preprocesing, the fovealization process has the effect of *stretching* the central region of the real image. It is for this reason that in Fig. 5 the obstacle seems to be very far from the center of the image when in fact it is not. The existance of a crash is coded in the data, the system however, detects the crash correctly just due to the activation of the bumber output neurons.

3.4 Error Measure

A Sobolev norm E_S is applied to the visual data to find an objective measure of the error, such that:

$$E_S = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 + \frac{1}{n-1}\sum_{i=1}^{n-1}(y_i - \hat{y}_i)^2 \qquad (1)$$

where $y_i = x_i - x_{i+1}$ and $\hat{y}_i = \hat{x}_i - \hat{x}_{i+1}$

The first term of Eq. 1 is the *Sum Squared Error (SSE)* between the real data x and the predicted data $\hat{x}$. The second term provides a measure not only of the difference between the two sets of data but of the posible existence of oscillation between the data and the prediction.

Fig. 7a shows the E_S for several trajectories during 40 time steps of Long Term Prediction (LTP). Fig. 7b shows the second term of Eq. 1 for the same period of time and the same trajectories, showing that the oscillations between real data and prediction are minimal. The curves on Fig. 7b stick to the general shape of their counterparts on Fig. 7a.

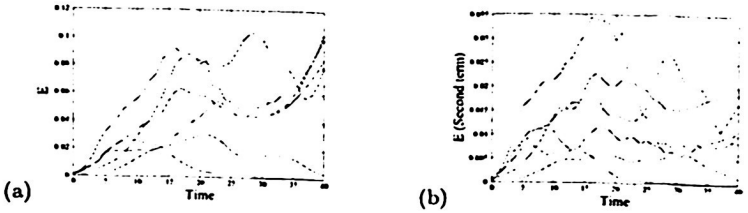


Fig. 7. E_S and the second term of Eq. 1 during 40 steps of LTP

The different shapes of the curves are due to the different starting points of the LTP as well as to the characteristics of the data. The two curves that tend to zero are predictions where no crash occurs, therefore the obstacles in the visual data dissappear.

4 Implementation

The trained forward model was implemented on the simulated agent used for collecting the data. The data coming from the camera was preprocessed and fed to the network as the agent was moving. The robot has the task of seeking a light source, avoiding collisions with nearby obstacles. The light sensing ability is completely independent of the vision or tactile senses of the robot. The robot, the light source and a number of objects are set on an initial random position.

Three behaviours are defined: a) light seeking, making the robot head directly into the light, b)prediction, if detecting high activation of the bumper units, the robot stops moving and performs an internal prediction and c) obstacle avoidance. The same thresholds are used; when there is activation on the bumper units long term prediction (LTP) is triggered. In case LTP finds the possibility of a future crash the obstacle avoidance behaviour takes over the light seeking behaviour.

As shown in Fig. 6 the activation of the bumper output units conveys information about the location of obstacles. This information is used to take avoiding action. Depending on the units that are registering the highest activation the robot decides where to turn. Fig. 8 shows the robot making its way through a set of obstacles on its way towards the light source. The robot performs a straight trajectory until the bumper units present high activation. The robot then performs an internal prediction which prompts it to change course.

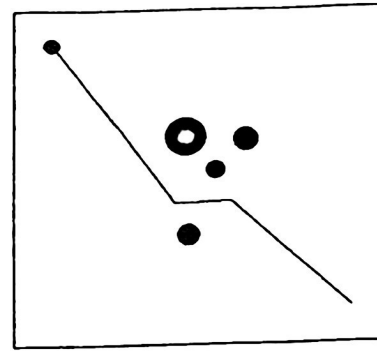


Fig. 8. Robot seeking a light source and avoiding a set of obstacles.

It is important to note that the robot can only predict sensory situations when the motor command has been constant for at least 3 steps. This means that once the robot turns, either by changing course towards the light or by avoiding an obstacle, it needs to move 3 steps with a constant motor command before being able to perform a prediction.

5 Conclusions

The presented experiments show that anticipation and forward models can provide agents with useful strategies. The agent presented here is capable of predicting the consequences of its own acts and take provisions for future actions. The system does this by learning an association between visual and tactile stimuli.

Our work differs from other research using forward models in several aspects. Hoffman et. al [11] presented a chain of forward models that provides an agent with the capability to select different actions to achieve a goal situation and perform mental transformations. The main difference with this system is that the forward model presented here performs the prediction of the same event (collision) using two sensor modalities (tactile and visual). The visual input to our forward model is completely grounded in that it does not have any sort of assumptions about distance. The system learns to estimate distance through interaction with its environment, associating visual input and tactile experience.

A form of anticipation is presented by Ziemke et al. [12], however their results for long term prediction are very constrained by the environment in which the robots evolved. In our case the model is capable of solving diferent scenarios, performing the necessary predictions and reacting timely to obstacles on its path.

Although the task presented here can be solved by different and simpler systems, this model presents the advantage of anticipation. The implementation of more complex tasks in the robot environment should prove that an anticipatory agent is capable of avoiding typical problems in which a reactive agent fails such as corners and dead-end situations.

6 Future Work

Further use of the system here presented can include behaviour in which the understanding of ego motion and its consequences is necessary. Current work, not reported here, includes analysis of the system predictions to distinguish between, changes brought up in the environment because of its own movement and those changes in the environment brought up by other agents and their respective actions.

The multi sensory representation of stimuli (visual and tactile) should also be present in the internal dynamics of the network. These dynamics should become more interesting with non-constant motor commands. As proposed by TEC [3], hidden units of the system would share resources when coding of actions and multi-sensory perception.

References

1. Witkowski, M.: Anticipatory learning: the animat as a discovery engine. In V., B.M., Gerard, P., O., S., eds.: Adaptive Behaviour in anticipatory learning systems (ABiALS'02), Edinburgh, Scotland (2002)
2. Butz, M.V., Sigaud, O., Gerard, P.: Internal models and anticipations in adaptive learning systems. In Butz M. V., Gerard, P., O., S., eds.: Adaptive Behaviour in anticipatory learning systems (ABiALS'02), Edinburgh, Scotland (2002)
3. Hommel, B., Müsseler, J., Aschersleben, G., Prinz, W.: The theory of event coding (TEC): A framework for perception and action planning. Behavioral and Brain Sciences 24 (2001) 849-937
4. Knoblich, G., Prinz, W.: Linking perception and action: An ideomotor approach. In Freund, H.J., Jeannerod, M., Hallett, M., Leiguarda, R.C., eds.: Higher-order motor disorders. Oxford University Press, Oxford, UK (2005)
5. Ziemke, T.: Rethinking grounding. In Riegler, A., Peschl, M., von Stein, A., eds.: Understanding Representation in the Cognitive Sciences. Kluwer Academic Publishers, New York (1999) 177-190
6. Kawato, M.: Bidirectional theory approach to integration. In Inui, T., McClelland, J., eds.: Attention and Performance XVI. MIT Press, Cambridge, MA (1996) 335 367
7. Blakemore, S.J., Frith, C.D., Wolpert, D.M.: Spatio-temporal prediction modulates the perception of self-produced stimuli. Journal of Cognitive Neuroscience 11(5) (1999) 551 559
8. Möller, R.: Perception through anticipation. A behavior-based approach to visual perception. In Riegler, A., von Stein, A., Peschl, M., eds.: Understanding Representation in the Cognitive Sciences. Kluwer Academic Publishers, New York (1999) 169-176

9. Dearden, A., Demiris., Y.: Learning forward models for robots. In: Int. Joint Conferences on Artificial Intelligence. Edimburgh (2005) 1440
10. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The rprop algorithm. (1993)
11. Hoffmann, H., R.Moeller: Action selection and mental transformation based on a chain of forward models. In: Proc. of the 8th Int. Conference on the Simulation of Adaptive Behavior, Cambridge, MA, MIT Press (2004) 213–222
12. Ziemke, T., Jirenhed, D.A., Hesslow, G.: Internal simulation of perception: a minimal neuro-robotic model. Neurocomputing **68** (2005) 85–104

Real World Applications

The Use of Artificial Neural Networks in Modeling the High-Pressure, Suspensive Waterjet Cutting Process of Syenite

Andrzej Perec[1], Grzegorz Dzuga[1], Mateusz Ruzylo[1]
[1] Koszalin University of Technology, Division of Computer Science Applications
Kwiatkowskiego Street 6E, 75-343 Koszalin, Poland
pereca@lew.tu.koszalin.pl
dzuga@wp.pl
zoltar@eranet.pl

(Paper received on August 12, 2006, accepted on September 27, 2006)

Abstract. This article presents the role of artificial neural networks in use of hydroabrassive suspensive jet cut process in syenite treatment. Three-ply layer perceptron type network with an error backpropagation learning algorithm was applied to describe this process. The article provides detailed description of neural network. This neural network simulates the treatment process and predicts its efficiency due to given parameters. The results were confronted with the laboratory results of complex studies on parameters of cutting syenite with a hydroabrasive suspensive jet, whose pressure is reduced to 30 MPa.

1. Introduction

In the recent years, high pressure water-jet machining has been competing effectively with conventional methods of separation of materials. This is above all owing to its universal nature as a result of wide-range possibilities such as cutting complex shapes, various materials or a possibility to conduct it in extreme conditions [1, 8] (hazard of fire or an explosion, work under water to 6000 m, etc.).

The most serious disadvantage of the so-far existing systems for cutting with a high pressure hydroabrasive jet and working at pressures of 400 MPa, is the use of an injector mixer to create the jet - due to its small efficiency, especially in the case of very big differences of working media velocities [2]. An elimination of an injector mixer and the use of the jet's circumferential motion [6] for mixing an initially created hydroabrasive mixture directly under a high pressure can result in a radical change of the situation. Similar machining effects are achieved even though the working pressure has been lowered even by an order of magnitude.

2. Test stand, method and material

The test stand has been constructed on the basis of BORJET 01 prototypical machinery. It is constructed the way that allows quick changes of hydraulic pipes, the mixing manner, and water supply to carry out an initially formed hydroabrasive jet [5].

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 173-180

BORJET 01 appliance has been built from two containers and four independent hydraulic branches (Fig. 1), which allow adjustment of the basic flow parameters [3]. 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 secures Borjet01 from damage made by to high pressure. It is set at the pressure of 30MPa.

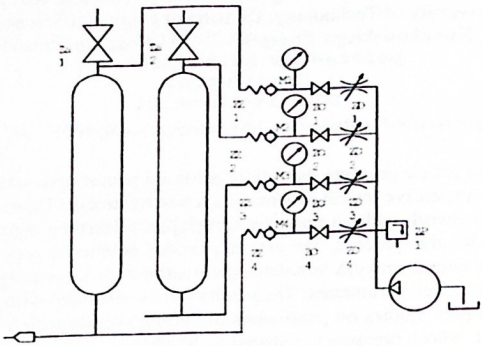


Fig. 1. Hydraulic diagram of device BORJET 01.

A hydraulic monitor P26 type is the source of a high pressure. It is made on the basis elements from a plunger pump made by 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.

Syenite is coarse-grained igneous rock, similar in appearance and composition to granite Fig. 2. Unlike granite, it contains very small quantity or even no quartz.

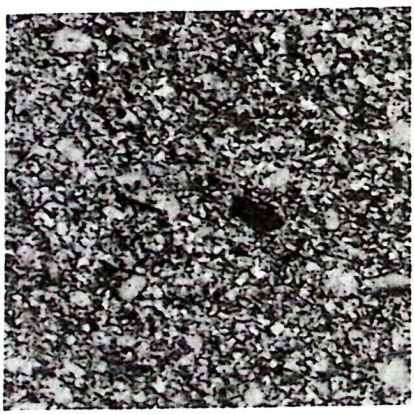


Fig. 2. The image of the polished surface of syenite.

Syenite is made mainly from feldspars, with mica, hornblende, and pyroxene. Varieties are distinguished (according to the ferromagnesian minerals contained) as augite

syenite, hornblende syenite, mica syenite, and nepheline syenite. Syenites are comparatively rare rocks, being found mainly in a few areas of the United States and Germany. They are used as substitutes of granites as building stones.

3. Artificial Neural Networks

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 the most important properties of the biological neuron. You should always remember that artificial equivalents functions are very simplified [9] 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. First block of adding up in which input signals are increased by suitable coefficients weights and added up then.

The topology of the net consisting from 5 neurons of the input layer, 5 neurons of hidden layer and one outputs neuron (Fig. 3) was accepted to modeling the waterjet cutting process [4].

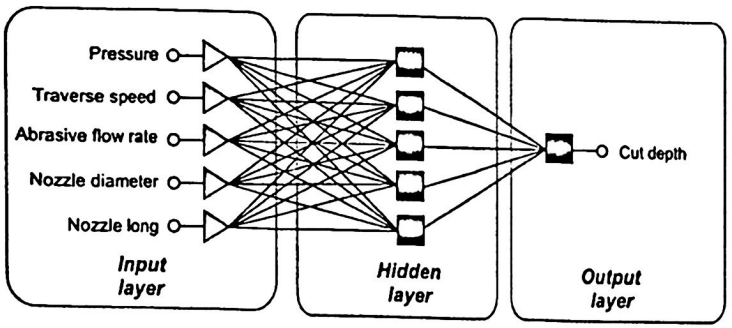


Fig. 3. Neural Network schematic diagram.

Input data to the input layer included- pressure, abrasive flow rate, size and diameter of nozzle and traverse speed. On the output layer cutting depth was given. 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 was taken from the accessible results of investigations directly. To learning process ware use 96 training cases that include both input and target output values [7]. 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 12000 iterations with learning rate of 0.03 and momentum 0.3 (Fig. 4).

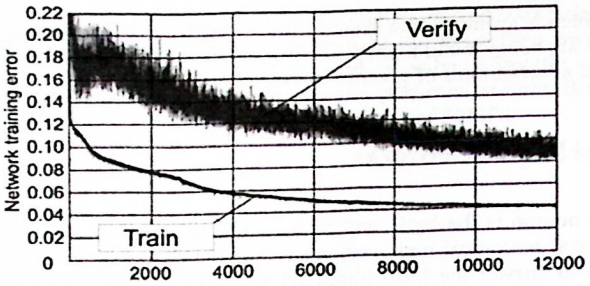


Fig. 4. Training error graph.

To research was utilized the commercial Statistica Neural Networks for Windows application of the StatSoft Inc company.

4. Effects of Artificial Neural Networks Modelling

Fig. 5(a) depicts the role of pressure and advance in syenite cut. In relation to the results obtained due to modelling, the maximum compatibility was observed with the advance at the maximum, whereas the biggest aberration (not beyond 15%) took place at the minimum pressure.

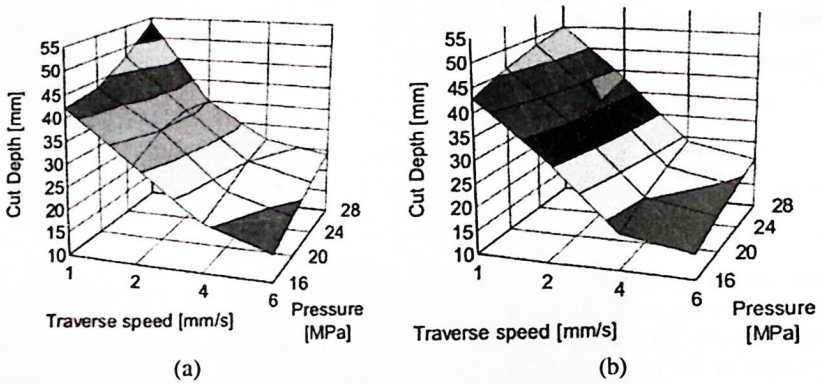


Fig. 5. Syenite cutting depth in a variable pressure and traverse speed conditions: a) laboratory analysis, b) modeled with the artificial neural networks.

Fig. 6(a) depicts how the pressure and abrasive flow rate influence the cutting depth. The compatibility with the results obtained in modelling (Fig. 6(b)), stays at the average of 1,56 mm. The maximum discrepancy, observed at the 28MPa pressure and 50 g/s abrasive flow rate, is at 4,67

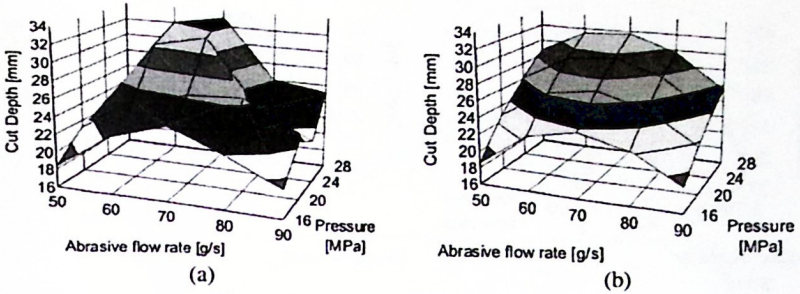


Fig. 6. Syenite cutting depth in a variable pressure and traverse speed conditions: a) laboratory analysis, b) modeled with the use of artificial neural networks.

The abrasive flow rate and the 50mm long nozzle diameter and their influence on cut depth are shown in Fig. 7(a). The average discrepancy of those results confronted with the results obtained in modelling (Fig. 7(b)) does not go beyond 2,5mm. The maximum discrepancy, not going beyond 6mm, was observed with aabrasive flow rate at minimum.

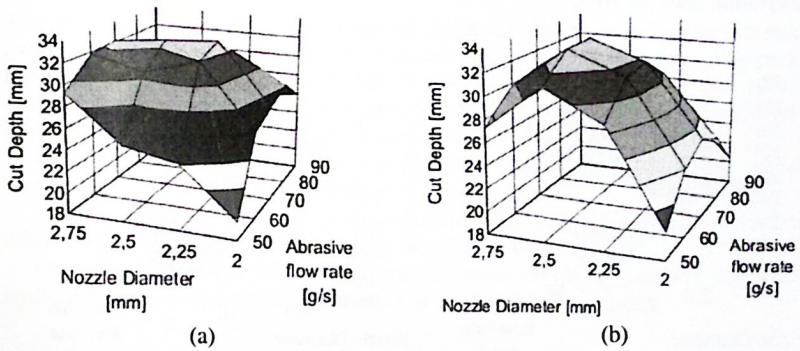


Fig. 7. Syenite cutting depth with the use of 50mm long nozzle in a variable nozzle diameter and abrasive flow rate conditions: a) laboratory analysis, b) modeled with the artificial neural networks.

Fig. 8(a) depicts how the abrasive flow rate and the 75mm long nozzle influence the cut depth results. The confrontation of those results with those obtained due to modelling (Fig. 8(b)) brought big compatibility, reaching 5%. The maximum discrepancy (not beyond 13%) was observed at the abrasive flow rate at minimum and the nozzle diameter of 2,25mm.

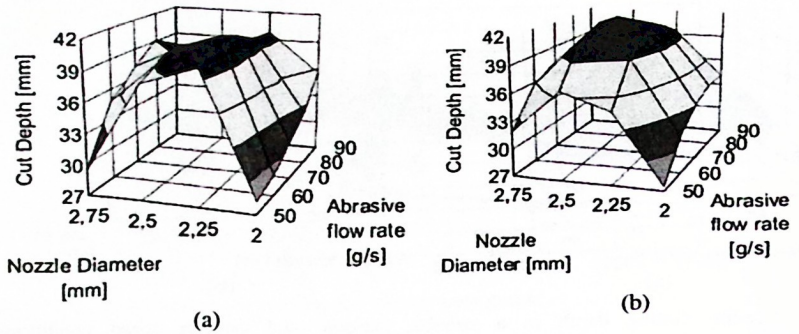


Fig. 8. Syenite cutting depth with the use of 75mm long nozzle in a variable nozzle diameter and a abrasive flow rate conditions: a) laboratory analysis, b) modelled with the artificial neural networks.

In Fig. 9(a) and Fig. 9(b) are shown the analogical relationships for 100mm long nozzle. Fig. 9. depict even more compatibility of real cut depth values with those obtained due to modeling. Only with the nozzle diameter and the abrasive flow rate at minimum the aberration goes for 10%.

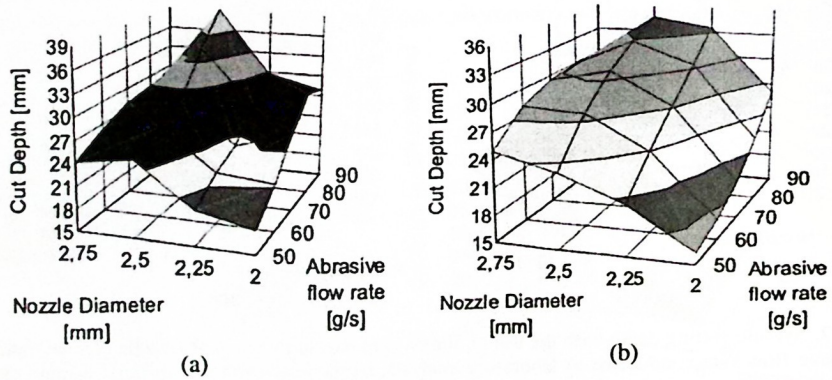


Fig. 9. Syenite cutting depth with the use of 100mm long nozzle in a variable nozzle diameter and a abrasive flow rate conditions: a) laboratory analysis, b) modelled with the artificial neural networks.

The overall comparison of lab and modeling results are shown in Fig. 10. Line ideal (y=x) illustrates how a model ideally matches real values. Points represent all values described in a report (values obtained with the use of ANN model)

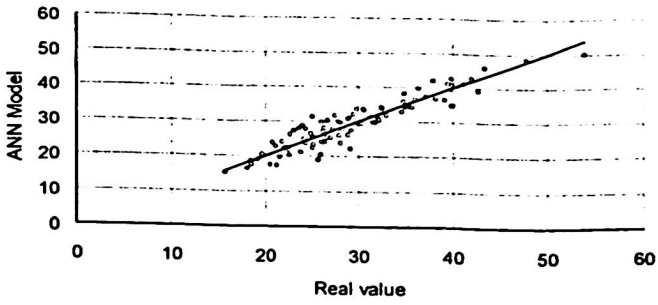


Fig. 10. Evaluation of matching Artifical Neural Network with measuring values.

Model to real values average discrepancy is 2,14mm (7,67%).

5. Summary

The artificial neural networks use in the cut depth designating, give similar estimates in every considered case. The divergences do not go beyond 8%. The remaining parameters modeling results do not exceed 5%. In some cases, the discrepancy is at 10%, and only one case resulted in 20% divergence. Standard discrepancy between modelled and laboratory-values are included in the interval from 2,16 to 3,24mm.

In most cases, the variation character due to the artificial neural network modeling was compatible with the results obtained in empirical way.

Making the neural network more complicated and choosing its parameters more adequately will result in the model being more "well-fitted".

Our next step is to use already trained artificial neural network to optimize hydroabrasive waterjet cutting parameters to maximize cutting depth.

References

[1] Brandt C., Louis H., Meier G., Tebbing G. (1994). Abrasive Suspension Jet at Working Pressure up to 200 MPa. 12th Int. Symposium on Jet Cutting Technology. Rouen, France.

[2] Chudy J., Perec A. (2001). Energetic Aspect of Highpresure Hydroabrasive Waterjet and Hydroabrasive, Suspensive Waterjet Machining Intensification. 2-nd Int. Conf. on Water Jet Machining WJM 2001. Cracow, Poland.

[3] Perec A. (2004). Some aspects of hydroabrasive suspensive jet cutting of syenite. 17th International Conference on Water Jetting - Advances and Future Needs. Mainz, Germany.

[4] Perec A., Dzuga G., Ruzylo M. (2006). The Ways To Utilize Artifical Neural Networks in Modelling The Highpressure Suspensive, Waterjet Cutting Process. Symposium on Mod-

elling and Simulation 2006. Lodz, Poland.

[5] Perec A. (2003). Some aspects of hydroabrasive suspensive jet cutting of marble. 2003 WJTA American Waterjet Conference. August 17-19, 2003 Houston, Texas, USA.

[6] Perec A. (1995). Influence of conditions highpressure, suspensive, hydroabrasive jet creation onto cutting materials parameters in lower working pressure aspect.(In Polish). Doctor trial.Technical University of Koszalin, Poland.

[7] Perec A., Dżuga G., Rużyło M. (2006). The use artifical neural networks in modelling the highpressure, suspensive waterjet cutting process of marble. 18th International Conference on Water Jetting. Gdańsk.

[8] Vijay M.M. (1989). Evalution of abrasive-entreined water jets for sloting hard rocks. 5th American Water Jet Conference. Toronto, Canada.

[9] Zurada J. (1992). Introduction to Artificial Neural Systems. West Publishing Company, St. Paul, New York, Los Angeles, San Francisco, USA.

Physical Model-Free Observer for Nuclear Reactors using Differential Neural Networks with Sliding Mode Learning

J. Humberto Pérez-Cruz and Alexander Poznyak
Department of Automatic Control, CINVESTAV-IPN, AP. 14740, Av. Instituto Politécnico Nacional No. 2508, C.P. 07360, México D.F., México. Tel.: +(525) 55 5061 37 41,Fax: +(525) 55 5061 39 82.e-mail: *jperez,apoznyak@ctrl.cinvestav.mx*
(Paper received on August 09, 2006, accepted on September 25, 2006)

Abstract. This paper investigates the problem of uncertain nonlinear state observation in a nuclear reactor when only the input and the output are available. The use of a differential neural network observer with sliding mode learning law is suggested. A very simplified model of the system is initially employed for off-line training of the neural network. When this process of training has finished, the observer can dispense with mathematical model completely and can realize the on-line state estimation within small margin of error despite uncertainty and noise. The efficiency of this technique is illustrated by simulation.

1. Introduction

Nuclear reactors are inherently nonlinear and very complex systems with time-varying parameters depending on a level power, fuel burnup, Xenon isotope production, among others factors. When, in this kind of systems, state observation is required, it is mostly used for two purposes: feedback control and fault detection. In these tasks, the observer employed has often been linear [1], [2]; consequently, its performance is only satisfactory in a small neighborhood of the operation point of the reactor. Nevertheless, if large variations of the system variables are presented, the previous option is not effective any more and some nonlinear state observer is required. In real situations, state estimation turns out to be a non trivial problem because, besides nonlinearity, variation of parameters, uncertainty and inclusive measurement noise must be considered. Moreover, even though parameters were constant, peculiarity of nuclear reactor model prevents its transformation into the, so-called, companion form; hence, common robust techniques such as high gain observers [3] can not be applied. Nonetheless, under these conditions, it is still possible to obtain an acceptable observation using others robust techniques among which it is worth mentioning neural networks (NN) and sliding modes (SM).

NN are an approach that has generated great enthusiasm as a consequence of their capability of functioning adequately despite a partially (or inclusive totally) incomplete information about plant model. NN could be classified as static (feedforward) or as differential (recurrent) ones [4]. In the first kind of networks, a system dynamics is approximated by a static mapping; therefore, the network outputs are uniquely determined by the current inputs and the weights. In contrast, differential neural networks (DNN) incorporate feedback in their structure. So, they overcome many problems associated with first ones such as global extrema search. Furthermore, DNN

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 181-193

have better properties of approximation.

In recent years, it has been proposed to use neural networks in nuclear reactors particularly for control [5-9] but generally the networks employed have been static or else, in many cases, they have lacked a rigorous proof of stability.

On the other hand, during the last two decades, SM have emerged as other powerful tool for control, identification, and estimation in uncertain environments. Basically, SM consist of the application of a discontinuous control action for reaching and maintaining the dynamics of a system on the, so-called, sliding surface. The major advantages of SM are: low sensitivity to plant parameter variations and disturbance, fast transient behavior, and exponential convergence [10]. However, despite these advantages, only a few applications of this technique have been reported in nuclear literature. So, in [11] a SM observer is used for estimating the external reactivity and the xenon concentration. In [12] is discussed a feedback controller based on SM observer for a space nuclear reactor.

In spite of fruitful research in DNN and SM, very few authors have considered the possibility of combining the advantages from these two techniques for obtaining an observer with better performance [13] and none, to our knowledge, has considered apply this "sliding neuro observer" to field of nuclear processes. Thereby, in this paper, it is suggested to estimate nuclear reactor states using a special kind of DNN which incorporate a switching correction term in their structure and also a sliding mode learning law. Since only the input and the output of reactor are available, a simplified, but imprecise, mathematical model of system is utilized for off-line initial training of DNN. When this process of training has finished, the observer can work without any mathematical model and can realize the on-line state estimation within small margin of error despite uncertainty and noise. The workability of suggested approach is illustrated by a simulation example.

2. Mathematical Model

In general, the nuclear reactor dynamics is described by the following so-called point kinetics equations with six delayed neutron precursor groups [14]:

$$\dot{n}_t = \frac{\rho_t - \beta}{\Lambda} n_t + \sum_{i=1}^{6} \lambda_i C_{i,t} \tag{1}$$

$$\dot{C}_{i,t} = \frac{\beta_i}{\Lambda} n_t - \lambda_i C_{i,t}, \quad i = 1,\ldots,6 \tag{2}$$

where n_t is the neutron power (W), $C_{i,t}$ is the power of the ith group delayed neutron precursor (W), ρ_t is the total reactivity, Λ is the effective prompt neutron lifetime (s), λ_i is the radioactive decay constant of ith group neutron precursor (s^{-1}), β_i is the fraction of ith group delayed neutrons, and β is the total delayed neutron fraction ($\beta=\sum_{i=1}^{6}\beta_i$). It is important to mention that the six group point kinetics equations (1) and (2) are in reality a set of seven ordinary differential equations; accordingly, their manipulation can result difficult. However, it is possible to reduce the system order by combining the

six precursor groups into an equivalent single group. First, let us define the effective precursor radioactive decay constant λ as

$$\lambda = \frac{1}{\beta}\sum_{i=1}^{6}\beta_i\lambda_i \tag{3}$$

Next, using (3), the equations (1) and (2) can be simplified into a second order system given by

$$\dot{n}_t = \frac{\rho_t - \beta}{\Lambda} n_t + \lambda C_t \tag{4}$$

$$\dot{C}_t = \frac{\beta}{\Lambda} n_t - \lambda C_t \tag{5}$$

where C_t is the equivalent power of all delayed neutron precursors.

Now then, the total reactivity has two components, the external reactivity $\rho_{ext,t}$ and the internal reactivity $\rho_{int,t}$, that is,

$$\rho_t = \rho_{ext,t} + \rho_{\text{int},t} \tag{6}$$

The external reactivity is related to the position of the control bars. Thus, the external reactivity is considered as the control input of the system. The relationship between the external reactivity and the bar position can be represented through an empirical static function. On the other hand, the internal reactivity is associated with the effects of the temperature feedback. These effects can be described [14] by

$$\dot{\rho}_{\text{int},t} = -\alpha K n_t + \alpha K n_0 - \gamma \rho_{\text{int},t} \tag{7}$$

where α the is negative temperature reactivity coefficient ($°C^{-1}$), K is the reciprocal of the reactor heat capacity ($°C/(W{\cdot}s)$), γ is the reciprocal of mean time for heat transfer to the coolant (s^{-1}), and n_0 is the initial power when the external reactivity is equal to zero. For suitability, we consider in this work that $n_0 = 1W$. The equations (4), (5), and (7) constitute a very simplified third order mathematical model of a nuclear reactor. The nominal parameters corresponding to a TRIGA MARK III research reactor located at National Institute of Nuclear Research of México [15] are as follows: $\alpha = 0.01359875\,°C^{-1}$, $\beta = 6.433x10^{-3}$, $\lambda = 0.4024s^{-1}$, $\Lambda = 38\mu s$, $\gamma = 0.2s^{-1}$, $K = 1/5.21045x10^{4}\,°C/(W{\cdot}s)$ whereas the ranges for the variables of the same reactor operating on standard conditions are: n_t from $1W$ to $1.1MW$, C_t from $420.72W$ to $462.79MW$, $\rho_{int,t}$ from -1.4354 to 0, $\rho_{ext,t}$ from 0 to 1.4354. Defining the state coordinates and the control input as $x_{1,t} := n_t$, $x_{2,t} := C_t$, $x_{3,t} := \rho_{int,t}$, $u_t := \rho_{ext,t}$, the equations (4), (5), and (7) can be represented in the standard state variable form

$$\begin{aligned}
\dot{x}_{1,t} &= -\frac{\beta}{\Lambda} x_{1,t} + \lambda x_{2,t} + \frac{1}{\Lambda} x_{1,t} x_{3,t} + \frac{1}{\Lambda} x_{1,t} u_t \\
\dot{x}_{2,t} &= \frac{\beta}{\Lambda} x_{1,t} - \lambda x_{2,t} \\
\dot{x}_{3,t} &= -\alpha K x_{1,t} + \alpha K n_0 - \gamma x_{3,t}
\end{aligned} \tag{8}$$

3. Observability Analysis

Consider the nonlinear nominal system described by (for the sake of briefness, system (9) is SISO but results presented in this work can be easily extended to MIMO systems)

$$\dot{x}_t = f(x_t, u_t, t),\ y_t = Cx_t \tag{9}$$

where $x_t \in \Re^n$ is the system state at time $t \geq 0$, $y_t \in \Re$ is the system output, $u_t \in \Re$ is the control action, $C \in \Re^{1\times n}$ is an output matrix and $f : \Re^n \times \Re \times \Re^+ \to \Re^n$.

Definition 1. *The system (9) is said to be observable within the interval time $[t_0, t_1]$, $t_1 > t_0$ when the output data $y(t)$ determine the initial state $x(t_0)$ completely.*

Likewise, let us define the extended output vector as

$$Y_t := \begin{bmatrix} y_t & \dot{y}_t & \cdots & y_t^{(n-1)} \end{bmatrix}^T \tag{10}$$

and the observability matrix

$$Q = \frac{\partial Y_t}{\partial x_t} \tag{11}$$

Results on observability of system (9) are presented in [16] and references there. These results can be summarized by the following:

Corollary 1. *The system (9) is* ***locally observable*** *in a neighborhood of the point x_t at time t, if*

$$\det Q \neq 0 \tag{12}$$

We will apply corollary 1 to (8) before trying to realize some observer design for the nuclear reactor. Since in this work it is considered that the only measurable state of (8) is $x_{1,t}$, then the system output is defined as $y_t := x_{1,t}$. Or else, in terms of the state vector, $y_t = \mathbf{C}\mathbf{x}_t$ where $\mathbf{C} := [1\ 0\ 0]$ and $\mathbf{x}_t \in \Re^3$. Next, let us calculate the corresponding extended output vector and observability matrix for (8)

$$Y_t = \begin{bmatrix} y_t & \dot{y}_t & \ddot{y}_t \end{bmatrix}^T = \begin{bmatrix} x_{1,t} & \dot{x}_{1,t} & \ddot{x}_{1,t} \end{bmatrix}^T$$

$$Q = \frac{\partial Y_t}{\partial x_t} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{\beta}{\Lambda} + \frac{1}{\Lambda} x_{3,t} + \frac{1}{\Lambda} u_t & \lambda & \frac{1}{\Lambda} x_{1,t} \\ \frac{\partial \dot{x}_{1,t}}{\partial x_{1,t}} & \frac{\partial \dot{x}_{1,t}}{\partial x_{2,t}} & \frac{\partial \dot{x}_{1,t}}{\partial x_{3,t}} \end{bmatrix}$$

In accordance with corollary 1, one can deduce that the nominal system (8) is not observable over the manifold

$$\det Q = \left(\frac{\lambda^2}{\Lambda} - \frac{\lambda\beta}{\Lambda^2} - \frac{\gamma\lambda}{\Lambda} \right) x_{1,t} + \frac{\lambda}{\Lambda^2} x_{1,t} x_{3,t} + \frac{\lambda}{\Lambda^2} x_{1,t} u_t + \frac{\lambda^2}{\Lambda} x_{2,t} = 0 \tag{13}$$

Remark 1. Notice that the manifold (13) is a two-dimensional surface in a 3-dimensional space and, hence, practically any dynamic trajectory (if the control is not a specially oriented for maintaining the dynamics within this manifold) will cross this manifold loosing the observability practically almost everywhere.

4. Differential Neural Network

4.1. Basic assumptions

The uncertain nonlinear system, which states will be reconstructed, is given by

$$\dot{x}_t = f(x_t, u_t, t) + \xi_{1,t}, \; y_t = C x_t + \xi_{2,t} \tag{14}$$

where x_t, y_t, u_t, f, and C are as in (9) but now the vectors $\xi_{1,t}$ and $\xi_{2,t}$ characterize mixed uncertainties that may include both unmodelled dynamics and deterministic disturbances. Notice that an alternative representation for (14) always could be

$$\dot{x}_t = A^{(0)} x_t + W^{(0)} \sigma(x_t) + B^{(0)} u_t + \tilde{f}_t \tag{15}$$

where the parameters $A^{(0)} \in \Re^{n \times n}$, $W^{(0)} \in \Re^{n \times n}$, $B^{(0)} \in \Re^{n \times 1}$ are subjected to adjustment, the activation vector-function $\sigma(\cdot) := [\sigma_1(\cdot), ..., \sigma_n(\cdot)]^T$ has sigmoidal components

$$\sigma_j(x) := a_{\sigma j} \left[1 + b_{\sigma j} \exp\left(- \sum_{j=1}^{n} c_{\sigma j} x_{j,t} \right) \right]^{-1} \text{ for } j = 1, \ldots, n \tag{16}$$

and $\tilde{f}_t := f(x_t, u_t, t) - A^{(0)} x_t - W^{(0)} \sigma(x_t) - B^{(0)} u_t + \xi_{1,t}$.

Hereafter it is supposed that the system (14), aside from observability condition given in corollary 1, comply with the following assumptions:

A) System (14) satisfies the (uniform on t) Lipschitz condition, that is,

$$f(x,u,t) - f(z,v,t) \le L_1 \|x - z\| + L_2 \|u - v\|$$
$$x, z \in \Re^n; \; u, v \in \Re^m; \; 0 \le L_1, L_2 < \infty$$

B) Admissible controls are bounded, $U^{adm} := \left\{ u : \|u\|_\Lambda^2 := u^T \Lambda u \le v_0 < \infty \right\}$, $\Lambda > 0$.

Besides, u_t is such that does not violate the existence of the solution to ODE (14).

C) There exists a control function $\mathring{u}_t$ belonging to the admissible control action set U^{adm} defined in assumption *B*, such that the uncertain nonlinear system (14) is quadratically stable, namely, there exists a Lyapunov function $V(x)$ such that $\frac{\partial V}{\partial x}\dot{x}_t \leq -n_1\|x_t\|^2$, $n_1 > 0$ whenever $u_t = \mathring{u}_t$.

D) The mixed uncertainties $\xi_{1,t}$ and $\xi_{2,t}$ are bounded, i.e., $\|\xi_{j,t}\|^2_{\Lambda_{\xi_j}} \leq \Upsilon_j$, $\Lambda_{\xi_j} > 0$, $j = 1,2$ (these matrices "normalize" the components to make possible to work with values of different physical nature). Besides, $\xi_{1,t}$ and $\xi_{2,t}$ do not violate the existence of the solution to ODE (14).

E) $A^{(0)}$ is Hurwitz and the pair $(A^{(0)}, C)$ is observable.

F) $\bar{f}_t$ is bounded, specifically, $\|\bar{f}_t\|^2_{\Lambda_f} \leq \bar{f}_0 + \bar{f}_1\|x_t\|^2_{\Lambda_{\bar{f}}}$, $\Lambda_f > 0$, $\Lambda_{\bar{f}} > 0$.

It is worth mentioning that the preceding assumptions are generally met for physically meaningful dynamic systems and a nuclear reactor is not exception.

4.2. Observer structure

Let us define DNN observer as follows:

$$\frac{d}{dt}\hat{x}_t = A^{(0)}\hat{x}_t + W_t\sigma(\hat{x}_t) + B^{(0)}u_t + K_1[y_t - C\hat{x}_t] + K_2\mathtt{sign}(y_t - C\hat{x}_t) \tag{17}$$

where $\hat{x}_t \in \Re^n$ is the estimated state, $W_t \in \Re^{m\times n}$ is the weight matrix and

$$sign(z) := \begin{cases} 1 \text{ if } z > 0 \\ -1 \text{ if } z < 0 \\ \in [-1,1] \text{ if } z = 0 \end{cases} \tag{18}$$

One can see that the structure of the observer (17) consists of three parts (see fig. 1):

- the neural network identifier with a single output layer
$$A^{(0)}\hat{x}_t + W_t\sigma(\hat{x}_t) + B^{(0)}u_t$$
- the Luenberger tuning term $K_1[y_t - C_t\hat{x}_t]$;
- the sign correction term $K_2\mathtt{sign}(y_t - C\hat{x}_t)$ which is intended to reduce the output external noise effect associated with real data. In general, this term improve the global performance of the observer particularly when the output error

$[y_t - C\hat{x}_t]$ is small and consequently the Luenberger term is not effective anymore.

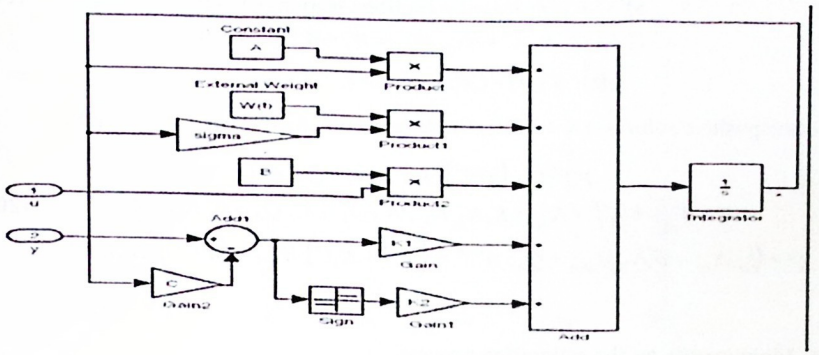


Fig. 1. Structure of DNN observer.

4.3. Off-line training of the network

Before using DNN observer (17) on-line, it is necessary to select adequate values for $A^{(0)}$, $B^{(0)}$, and $W^{(0)}$. This preliminary process of selection is known as off-line training of DNN. If the off-line knowledge of all states of (14) is not available then at least it is necessary to resort to a simplified and inclusive imprecise model of (14) as the generator of N training data $\left(u_{t_k}, x_{t_k}\right)|_{k=1,N}$. Basically, the training process consists of two stages: First, using "try-to-test" method, values for $A^{(0)}$ and $B^{(0)}$ are proposed such that assumption E is satisfied. Next, using a least square method, the best nominal value of $W^{(0)}$ is determined. Let us define this value as $W^{(0)*} = \min \sum_{k=1}^{N} \left\| \mathcal{T}_{t_k} \right\|^2 = \left[Y_{t_1} \ \cdots \ Y_{t_N}\right] \left[\sigma\left(x_{t_1}\right) \ \cdots \ \sigma\left(x_{t_N}\right)\right]^+$ where $Y_{t_k} := \dot{x}_{t_k} - A^{(0)} x_{t_k} - B^{(0)} u_{t_k}$ (here $[\cdot]^+$ means pseudoinverse in Moore-Penrose sense). Starting from t_N, we are ready to initiate the learning (or DNN adaptation) process.

4.4. Learning law

The weight matrix W_t is adjusted by the the following learning law:

$$
\begin{aligned}
&W_t^{(i,j)} = -k\mu_t S_t^{(i,j)} \operatorname{sign}\left(W_t^{(i,j)}\right),\ i,j=\overline{1,n} \\
&\mu_t := N_\delta P W_t^T \sigma(\hat{x}_t)_{\Pi}^2 + 2e_t^T C N_\delta P W_t \sigma(\hat{x}_t) \\
&S_t^{(i,j)} = 1/n,\ i,j=\overline{1,n} \text{ (uniform learning)}, k>0 \\
&\Pi := C^T \Lambda_{\xi_2} C + \delta \Lambda_1,\ W_t := W_t - W^{(0)*},\ \delta > 0 \\
&e(t) := y(t) - C\hat{x}(t),\ N_\delta := \left(C^T C + \delta I\right)^{-1}
\end{aligned}
\tag{19}
$$

P is the positive solution (if it exists) for the algebraic Riccati equation given by

$$
\begin{aligned}
&P\bar{A}^{(0)*} + \left(\bar{A}^{(0)*}\right)^T P + PRP + Q = 0 \\
&R = W_{\Lambda_\sigma^{-1}} + \Lambda_f^{-1} + \Lambda_{\xi_1}^{-1} + K_1 \Lambda_{\xi_2}^{-1} K_1^T,\ \Lambda_f \bar{f}_1 \le n_1,\ Q_0 > 0,\ \Lambda_\sigma > 0 \\
&Q = \left(l_\sigma \Lambda_\sigma + \delta \Lambda_1\right) I_{n\times n} + Q_0,\ \bar{A}^{(0)*} = \left(A^{(0)} + K_1 C\right),\ W_{\Lambda_\sigma^{-1}} = \left(W^{(0)*}\right)^T \Lambda_\sigma^{-1} W^{(0)*}
\end{aligned}
\tag{20}
$$

4.5. Main results on the estimation process

One of the principles advantages of (17) and of the corresponding learning law (19), it is the possibility of guaranteeing that both averaged estimation error and weights are upperly bounded. In fact,

Theorem 1. *If there exist positive definite matrices* $\Lambda_f, \Lambda_{\xi_1}, \Lambda_{\xi_2}, \Lambda_\sigma, \Lambda_1, Q_0$ *and positive constants* δ, k *such that the matrix Riccati equation (20) has a positive definite solution, then the DNN observer (17), supplied by the learning law (19), with any matrix* K_1 *guarantying that* $\bar{A}^{(0)*}$ *is stable, that is,* $\bar{A}^{(0)*} := \left(A^{(0)} + K_1 C\right)$ *is Hurwitz and* $K_2 = \lambda P^{-1} C^T,\ \lambda > 0$ *provides the following upper bound for the averaged estimation error*

$$
\begin{aligned}
&\lim_{T\to\infty} \frac{1}{T} \int_{t=0}^{T} \|\Delta_t\|_P^2 dt \le \lim_{T\to\infty} \frac{1}{T} \int_{t=0}^{T} \left(\|\Delta_t\|_P^2 + 2\lambda \sum_{i=1}^{n} C\Delta(t)_i \right) dt \le \frac{\rho_Q}{\alpha_Q} \\
&\rho_Q := \bar{f}_0 + \Upsilon_1 + 2\Upsilon_2 + 4\lambda\sqrt{n\Upsilon_2},\ \alpha_Q := \lambda_{\min}\left(P_1^{-1/2} Q P_1^{-1/2}\right) > 0
\end{aligned}
\tag{21}
$$

Lemma 1. *For the learning law (19) and for any* $\rho \in (0, \lambda_{\min}(\Pi))$ *the following property holds:*

$$
\lim_{T\to\infty} \frac{1}{T} \int_{t=0}^{T} \|W_t \sigma(\hat{x}_t)\|^2 dt \le \frac{\lim_{T\to\infty} \frac{1}{T} \int_{t=0}^{T} \|C^T e_t\|^2_{(\Pi-\rho I)^{-1}} dt}{\rho \lambda_{\min}\left(P N_\delta^2 P\right)}
\tag{22}
$$

The proofs of theorem 1 and lemma 1 are achieved by means of Lyapunov-like analysis in [17].

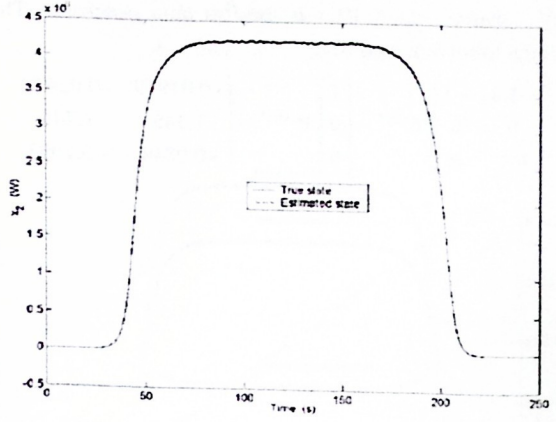


Fig. 2. The estimation process for the state $x_{2,t}$.

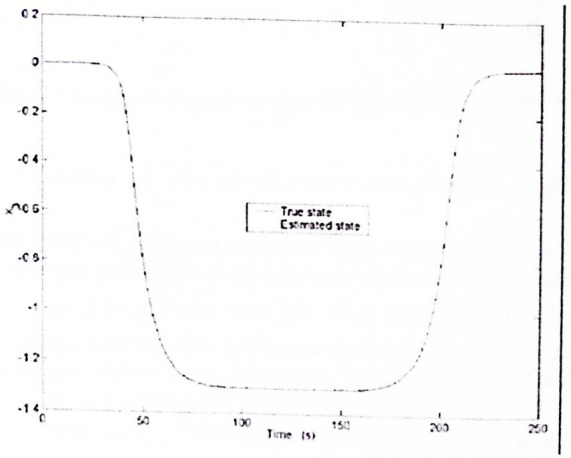


Fig. 3. The estimation process for the state $x_{3,t}$.

5. Numerical Example

In this example the state estimation process of a nuclear reactor via the DNN observer (17) is illustrated. In first term, to overcome the measuring complexity or else the absence of adequate sensors that prevent the off-line knowledge of all states, *the model* (8) *with nominal parameters given in the section 2 and with nominal initial condition*

$x_0 = [n_0, \frac{\beta}{\lambda\Lambda} n_0, 0]^T$ *where* $n_0 = 1W$ *is used as data generator.* The preliminary training produces the following results:

$$A^{(0)} = \begin{bmatrix} -1.2 & -1.3 & -1 \\ 1 & 0 & 0 \\ 0 & 0.1 & 0 \end{bmatrix}, B^{(0)*} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, W^{(0)*} = \begin{bmatrix} -0.01919 & -0.02472 & 0.1183 \\ 1.345 & 1.744 & 0.2911 \\ -0.0844 & -0.1093 & -0.02583 \end{bmatrix}$$

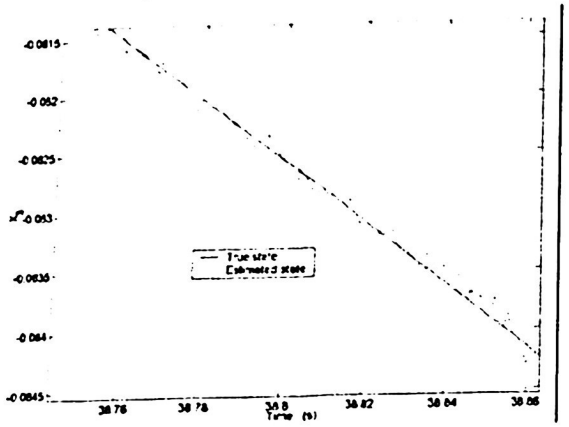


Fig. 4. The estimation process for the state $x_{3,1}$ (close-up).

To show the robustness properties of the technique presented, the DNN observer (17) is proved when the plant model (8) is simulated on the following new conditions: First, the initial condition is changed such that now $x_0' = [n_0, 10\frac{\beta}{\lambda\Lambda} n_0, 1\times 10^{-6}]^T$ where $n_0 = 1W$. Second, to model some effects associated with real instrumentation, the plant output is polluted with a uniformly distributed noise which magnitude is always approximately 3 % respect to plant output with independence of its level. Third, the parameter values of (8) are changed to $\alpha = 0.0097°C^{-1}$, $\beta = 0.0072$, $\lambda = 0.3942s^{-1}$, and $\Lambda = 30\mu s$ (both γ and K stay equal). Although, apparently these changes are small, as it will be seen a Luenberger observer basis on (8) with nominal values of the section 2 is not able to work satisfactorily on these new conditions (of course, these changes are supposed to be unknown for the DNN observer). By "try-to-test" method, the parameters of DNN observer (17) are selected as follows:

$$K_1 = [-2.7, 15, -0.1]^T, \; K_2 = [0.054, -0.037, 0.079]^T, \; k = 1.2 \text{ and}$$

$$P = 10\begin{bmatrix} 9 & 4 & -3 \\ 4 & 10 & 2 \\ -3 & 2 & 3 \end{bmatrix}.$$

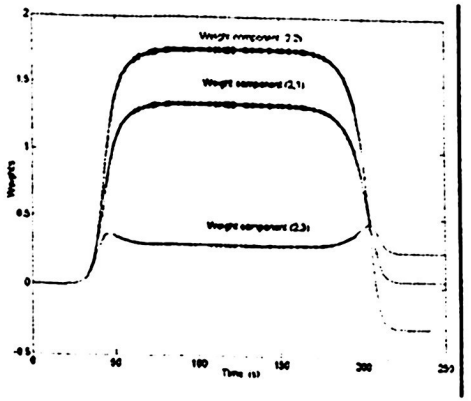


Fig. 5. Time evolution of weights $W_t^{(2.1)}$, $W_t^{(2.2)}$ and $W_t^{(2,3)}$.

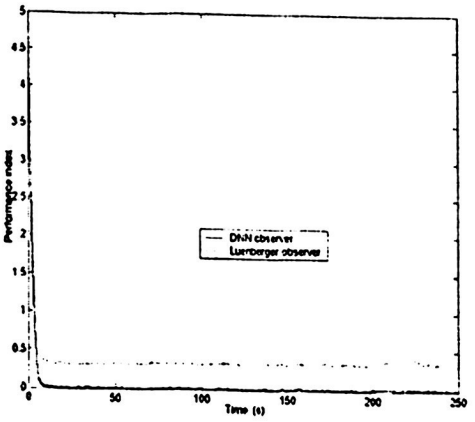


Fig. 6. Performance index for global estimation process.

The results of estimation process are displayed in Figs. 2-6. In Fig. 2 the estimated state masks almost completely to the true state $x_{2,t}$. In Fig. 3 the difference between true state $x_{3,t}$ and estimated state $\hat{x}_{3,t}$ is practically imperceptible. However, a closer look

provided by Fig. 4 illustrates better such difference. Some elements of weight matrix are presented in Fig. 5. It is important to mention that due to wide range of state values associated with (8) it is necessary to resort to the normalization of variables in both the preliminary training and on-line estimation process. Such normalization does not affect the estimation results. Instead, it permits to the observer (17) works satisfactorily. To quantify the global performance of an observer, let us define the following performance index:

$$J_t = \frac{1}{t+\varepsilon}\int_{s=0}^{t} \left\| x_s - \hat{x}_s \right\|_{Q_0}^2 ds, Q_0 > 0, \varepsilon = 0.01. \tag{23}$$

Performance indexes of the observer studied here and a Luenberger observer are shown in Fig. 6 for comparison.

6. Conclusions

We have studied the use of DNN observers with SM learning law for nuclear reactor state estimation in the presence of measurement noise and uncertainty. As it was seen in the numerical example, this technique represents a significant advantage respect to a simple Luenberger observer. Since it permits the possibility of utilizing reduced order systems besides guaranteeing the averaged boundness of both the DNN weights and the estimation error, we conclude that this methodology is promising in nuclear reactor applications and may constitute the basis for posterior development of efficient feedback controller designs.

References

[1] H. Arab-Alibeik and S. Setayeshi, Improved temperature control of a PWR nuclear reactor using an LQG/LTR based controller, *IEEE Trans. Nucl. Sci.*, vol. 50, no. 1, pp. 211-218, Feb 2003.

[2] Q. Li and J. A. Bernard, Design and evaluation of an observer for nuclear reactor fault detection, *IEEE Trans. Nucl. Sci.*, vol. 49, issue 3, part 2, pp. 1304-1308, Jun 2002.

[3] A. Tornambè, Use of asymptotic observers having high-gains in the state and parameter estimation, in Proc. of the 28th Conf. on Decision and Control, Tampa, Florida, 1989, pp. 1791-1794.

[4] S. Haykin, *Neural Networks: A Comprehensive Foundation*, IEEE Press, NewYork, 1994.

[5] N. S. Garis, I. Pázsit, U. Sandberg, and T. Andersson, Determination of PWR control rod position by core physics and neural network methods, *Nucl. Technol.*, vol. 123, pp. 278-295, Sep. 1998.

[6] K. Nabeshima, E. Ayaz, S. Seker, B. Barutcu, E. Turkcan, and K. Kudo, Nuclear Power Plant Monitoring with MLP and RBF Network, *Proc. of the 6th International FLINS Conference on Applied Computational Intelligence*,

Blankenberge, Belgium, September, 2004.

[7] M.G. Na, S.H. Shin, D.W Jung, S.P Kim, J.H. Jeong, and B.C. Lee. Estimation of Break Location and Size for Loss of Coolant Accidents Using Neural Networks, *Proc. of the 6th International FLINS Conference on Applied Computational Intelligence*, Blankenberge, Belgium, September, 2004.

[8] M. N. Khajavi, M. B. Menhaj, and A. A. Suratgar, A neural network controller for load following operation of nuclear reactors, in Proc. Int. Joint Conf. on Neural Networks, Washington, DC, July 2001, pp. 491-496.

[9] M. Boroushaki, M. B. Ghofrani, C. Lucas, and M. J. Yazdanpanah Identification and control of a nuclear reactor core (VVER) using recurrent neural networks and fuzzy systems, *IEEE Trans. Nucl. Sci.*, vol. 50, no. 1, pp. 159-174, Feb. 2003.

[10] V. I. Utkin, "*Sliding Modes in Control Optimization*," Springer-Verlag, Berlin, 1992.

[11] P. Wang, T. Aldemir, and V. I. Utkin, Estimation of xenon concentration and reactivity in nuclear reactors using sliding mode, in Proc. of the 40th Conf. on Decision and Control, Orlando, Florida, Dec. 2001, pp. 1801-1806.

[12] Y. B. Shtessel, Sliding mode control of the space nuclear reactor system, *IEEE Trans. Aerosp. Electron. Syst.*, vol. 34, no. 2, pp. 579-589, Apr. 1998.

[13] I. Chairez, A. Poznyak and T. Poznyak, "Model Free Sliding Mode Neural Observer for Ozonation Reaction", *Proc. of the 8-th Int. Workshop on Variable Structure Systems*, Villanova i la Geltrú, Spain, 6-8 September, 2004

[14] D. L. Hetrick, Dynamics of Nuclear Reactors, Editorial The University of Chicago Press. 1971.

[15] J. Viais, Cálculo de los Parámetros Fundamentales para el Estudio Dinámico del Reactor TRIGA Mark III del Centro Nuclear de México, B. Sc. Thesis, Universidad Nacional Autónoma de México, 1994.

[16] A. Sabanovic, Deterministic output noise effects in sliding mode observation, in Variable Structure Systems: From Principles to Implementation, Institution of Electrical Engineers, 2004, ch. 3..

[17] I. Chairez, A. Poznyak and T. Poznyak, New Sliding Mode Learning Law for Dynamic Neural Network Observer, *IEEE Trans. Circuits Syst. II, to appear.*

Unusual Changes of Consumption Detection in Mobile Phone Users

Paola Britos, Hernan Grosser, Enrique Sierra, Ramon Garcia-Martinez
Software & Knowledge Engineering Center (CAPIS) Graduate School.
Buenos Aires Institute of Technology. Av. Madero 399. (1106) CABsAs. Argentina.
Intelligent Systems Laboratory. School of Engineering.
University of Buenos Aires. P. Colón 850 4to Piso. Ala Sur. (1063) CABsAs. Argentina.
rgm@itba.edu.ar

(Paper received on September 01, 2006, accepted on September 26, 2006)

Abstract. The points being approached in this paper are: the problem of detecting unusual changes of consumption in mobile phone users, the corresponding building of data structures which represent the recent and historic users' behaviour bearing in mind the information included in a call, and the complexity of the construction of a function with so many variables where the parameterization is not always known.

1. Introduction

The preparation of manuscripts that are to be reproduced by photo-offset requires special care. Papers submitted in a technically unsuitable form will be returned for retyping, or canceled if the volume cannot otherwise be finished on time. When a mobile call is started, the cells or switches record that it is being made and they produce information referring to this event. These records are commonly called CDR's (Call Detail Records). CDR's contain useful information about the call so that it can be properly charged to whom it may correspond [1]. They can also be used to detect any fraudulent activity considering well-studied fraud indicators. That is, processing an amount of recent CDR's and comparing a function of the different fields such as, IMSI (International Mobile Subscriber Identity, which univocally identifies a user in a mobile phone network), date of call, time of call, duration, type of call (with a specific criteria). If this function retrieves a value that is considered beyond normal limits, an alarm is set off. This alarm must be taken into account by fraud analysts in order to determine if there has been any activity in bad faith or not. To be able to process these CDR's, it is necessary to make previously a process known in telecommunications as mediation, in which the information is read with the format of record in which CDR's come and then it is encoded in a new format of record which is understood by the fraud system.

The existing systems of fraud detection try to consult sequences of CDR's by comparing any field function with fixed criteria known as Triggers. A trigger, when activated, sends an alarm that leads to fraud analysts' investigation. These systems make what is known as a CDR's absolut analysis and they are used to detect the extremes of fraudulent activity. To make a differential analysis, patterns of behavior of the mobile phone are monitored by comparing the most recent activities to the historic

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 195-204

use of the phone; a change in the pattern of behavior is a suspicious characteristic of a fraudulent act. [1]

2. Description of the problem

In order to build a system of fraud detection based on a *Differential analysis* it is necessary to bear in mind different problems that arise and must be carefully worked on. These are:

2.1. The problem of building and maintaining "users' profiles"

The majority of fraud indicators are not analysed by using a unique CDR. In a system of differential fraud detection, information about the history together with samples of the most recent activities is necessary. An initial attempt to solve the problem could be to extract and encode CDR's information and store it in a given format of record. To do this, two types of records are needed; one, which we shall call CUP (Current User Profile) to store the most recent information, and another, to be called UPH (User Profile History) with the historic information [4, 5, 6]. When a new CDR of a certain user arrives in order to be processed, the oldest arrival of the UPH record should be discarded and the oldest arrival of the CUP should enter the UPH. Therefore, this new, encoded record should enter CUP. This information should be stored in a compact form so it is easy to analyze later on by the system of fraud detection. Considering the amount of information that a CDR contains it is necessary to find a way to "classify" these calls into groups or prototypes where each call must belong to a unique group. This raises several important questions to deal with: (a) What structure must CUP and UPH records have? (b) How many groups or prototypes must CUP and UPH records have in order to take the necessary information?, (c) How can calls be classified in the different, pre-defined prototypes? and (d) How to encode calls so that they can be "prototyped".

2.2. The problem of detecting changes in behavior

Once the encoded image of the recent and historic consumption of each user is built, it is necessary to find the way to analyze this information so that it detects any anomaly in the consumption and so triggers the corresponding alarm. It's here that the most important question of the whole paper arises and it is: How can the change in a user's pattern of behavior be detected? Our problem, then, is focused not only on the detection of abnormal changes in consumption, but also and fundamentally on building the data structures that represent the recent and historic behavior of each user considering the great amount of information that a call takes and the complexity of building a function with so many variables of input, complex and unknown. Having analyzed all the problems in question, we can make a graphic summary of what the fraud system should be like (Fig 1).

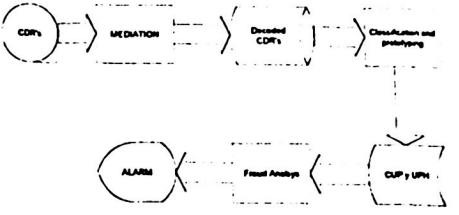


Fig. 1. Graphic representation of the problem to be solved.

3. Description of the suggested solution

The solution that has been developed has taken into account each and every question mentioned before, attempting to solve them in the most effectual and effectively possible way. Below is the presentation of each answer to the questions met in the analysis of the problem. In order to be able start processing the CDR's, a new format of record (mediation process output) must be created containing the following information: IMSI, date of call in YYYYMMDD format, time of the call in HH24MISS format, duration of call in 00000 format and type of call classified as LOC (local call), NAT (national call) and INT (international call). With this information together with the necessary data, it is possible to start solving the following and most important questions by using as input data the output of mediation process.

3.1. Solution to the construction and maintenance of "user's profiles"

The first point to solve is to determine how to make the CUP and UPH profiles. This means fixing the patterns that will make up each of the profiles. The patterns must have information about the user's consumption, separating LOC consumption (local calls), NAT (national calls) and INT (international calls) respectively. An interesting way to build these patterns is using neural networks so as to determine the space of all users' calls generating a space of patterns which represent the consumption of all users, and then generating a distribution of frequencies by user in which the probability of a user making calls following this pattern is represented [4]. To sum up, when a user's profile is built, the representation of the distribution of frequency in which a certain user makes a certain call is made. This data structure shows the user's pattern of consumption. Among other advantages, neural networks have the capacity to classify the information in certain patterns. Especially, SOM (Self Organizing Map) networks can take this information and build these patterns in a way that is not supervised by similarity criteria, and without knowing anything a priori about the data [8, 7, 3]. In our case, all the calls made by all users can be processed so that the networks, depending of the quantity of calls there are of each type, generate the patterns (creating resemblance groups) that represent all of them. To avoid noise in the data, three neural networks are

used to generate patterns to represent LOC, NAT, and INT calls respectively. The user's profile is built using all three patterns generated by the three networks. The data used to represent a pattern are the time of the call and its duration. We know that if we represent, in a Cartesian axis, the time of all calls and their corresponding duration, we will obtain a rectangle full of points. The idea is to obtain a graph in which only the most representative points of the whole space will appear; that is the neural network task. Once the patterns that will be used to represent the user's profile are obtained, it is necessary to start filling them with information. The procedure consists of taking the call to be analyzed, encoding it and letting the neural network decide which pattern it resembles. After getting this information, the CUP user profile must be adapted in such a way that the distribution of frequency shows that the user now has a higher chance of making this type of calls. Knowing that a user's profile has K patterns that are made up of L patterns LOC, N patterns NAT and I patterns INT, we can build a profile that is representative of the processed call and then adapt the CUP profile to that call. If the call is LOC, the N patterns NAT and the I patterns INT will have a distribution of frequency equal to 0, and the K patterns LOC will have a distribution of frequency given by the equation [4].

$$v_i = \frac{e^{-\|X-Q_i\|}}{\sum_{j=1}^{L} e^{-\|X-Q_j\|}}$$

where:
X: encoded call to be processed
v : probability that X call could be i pattern
Qi: pattern i generated by the neural LOC network.

Notice that:

$$\sum_{j=1}^{K} v_j = 1$$

If the call were NAT, then L must be replaced by N and the distribution of LOC and INT frequencies will be 0; if the call were INT, then L must be replaced by I and the distribution of LOC and NAT frequencied will be 0.

Then, we can define the vector that represents V call, of K dimension, as

Vi = vi, with $1 \le i \le L$
Vi = 0, with $L+1 \le i \le K$, when the call is LOC.

Vi = vi, with $L+1 \le i \le L+N$
Vi = 0 with $1 \le i \le L$ y $L+N \le i \le K$, when the call is NAT.

Vi = vi, with $L+N+1 \le i \le K$
Vi = 0, with $1 \le i \le L+N$, when the call is INT.

Now that we have V vector, we can adapt CUP vector with the information of the processed call:

$CUP_i = \alpha_{LOC} CUP_i - (1-\alpha_{LOC})V_i$, with $1 \le i \le K$, when the call is LOC,

$CUP_i = \alpha_{NAT} CUP_i - (1-\alpha_{NAT})V_i$, with $1 \le i \le K$, when the call is NAT,

$CUP_i = \alpha_{INT} CUP_i - (1-\alpha_{INT})V_i$, with $1 \le i \le K$, when the call is INT, where

α_{LOC}: adaptability rate applied when call X is incorporated to CUP, if X corresponds to a local call.

α_{NAT}: adaptability rate applied when call X is incorporated to CUP, if X corresponds to a national call.

α_{INT} adaptability rate applied when call X is incorporated to CUP, if X corresponds to an interantional call.

Once the CUP profile is adapted, it is compared with the UPH profile and then it is decided whether there has been a significant change in behaviour (engine of detection of changes in behaviour). After this, the UPH is adapted with the CUP information, only if the number of calls necessary to change the historic patterns has been processed.

$$UPH_i = \beta UPH_i + (1-\beta)CUP_i.$$

with $1 \le i \le K$, where

β: adaptability rate applied when CUP is incorporated to UPH.

3.2. Solution to the detection of changes in behavior

In order to settle whether there have been changes in the pattern of behavior or not, it is necessary to compare, somehow, the CUP and UPH profiles and decide if the difference between them is big enough so as to set an alarm off. Because both the CUP and the UPH are vectors that represent frequency distributions, a vectorial distance can be used to compare how different they are. For this, the Hellinger distance (H) can be used; it indicates the difference between two distributions of frequency [4]. This distance will always be somewhere between zero and two, where zero is for equal distributions and two represents orthogonality. The value of H will establish how different must CUP and UPH frequency distributions be, in order to set an alarm going. By changing this value, there will be more or fewer alarms set off.

3.3. Limitations of the solution

This solution is focused, as we described, on the analysis of the user's differential consumption. One case that may not be detected would be that in which the user *always makes a lot of calls of the same type with a high consumption*, as his pattern of behavior would never change. That is why there should always be a combination of several solutions in order to have a system of fraud detection that can detect different types of fraud. In this case, the absolute analysis would be a good solution. The other limitation centres in that the patterns are static, so that if the way in which the company users consume changes completely, it will be necessary to train again the neural networks to establish new patterns that represent the total space of calls and to re- build the CUP and UPH profiles as from the new distributions.

4. Experimentation

4.1. Methodology used

The experiments were divided in two parts: the first was focused on the training of the neural network and the generation of patterns to build the user's profiles later on; the second was aimed at the analysis of the calls made by high-consumption users and the corresponding analysis and detection of alarms. The second part of the test was divided again into two different experiences: 1) updating of UPH profile with each call (f= 1 call) and low Hellinger threshold (H) for the setting off of alarms of change of behavior; 2) updating of UPH profile once a day (f=1 day) and high Hellinger threshold (H).

4.2. Experiments on the generation of patterns

Three Self-Organizing Map (SOM) neural networks were built for the generation of patterns for LOC, NAT and INT calls respectively. Each of the networks was trained with an amount of calls that was representative of the consumption that company users made during a couple of days at all times. The calls were introduced to the network in a disorderly manner so that the patterns that were generated were not representative only of the time and duration of the last calls. The result of this experience defined the patterns to build the users' profiles. The patterns are made up of the time of the call and its duration in minutes, which managed to build a discrete space composed of all the types of called made by any user in a fixed quantity representative of that space.

4.3. Experiments on the construction of profiles and detection of behaviours

Once the patterns that define the space of all calls are obtained, tests have been carried out on the construction of user's profiles through the development of a distribution of frequencies of each of the patterns for each profile (CUP and UPH) and the corresponding detection of alarms. The process was based on the introduction to the system of calls made within a period of three months by users reported as "high-consumption user". With each call the CUP user profile was updated, it was then compared with the UPH profile, thus, obtaining the Hellinger distance between them. If it surpassed the fixed threshold, an alarm was set off. Depending on the parameter of updating frequency of UPH profile (f), the UPH was updated with the corresponding contribution of the CUP. At the moment of inputting a user's first call, all CUP and UPH patterns were initialized with the same distribution of frequency, assuming a priori that the user had the same tendency to make any type of call, without any information. Moreover, this experience was carried out twice; the first one updating the UPH with each call, therefore, with a low Hellinger threshold (H) for the detection of alarms. This was because the difference that may arise between the CUP and UPH profiles was too small if updating the historic profile with each call, due to the fact that the historic profile tended to be the same as the current profile. The second experience

was made by updating the UPH once a day and a high Hellinger (H) threshold to detect important differences that can be considered as changes in behavior.

5. Results

5.1. Generation of patterns

In this section results are presented after the training of the three neural networks. This means, the results show each of the patterns that the networks fixed as most representative of the space of all the users' calls. Three graphs are represented (one for each network) to show the patterns that were generated. On axis X, the time of the call is shown and on axis Y, the duration expressed in minutes is illustrated. Each of the points represented corresponds to a pattern being chosen by the network as representative of the sample. In the local neural network graph, 144 patterns are shown, in the NAT network, 64 and in the INT network, 36.

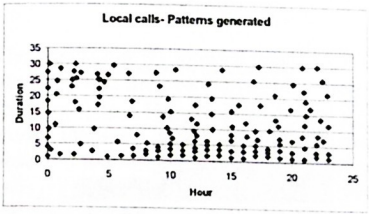


Fig. 2. Patterns generated after the training corresponding to local calls

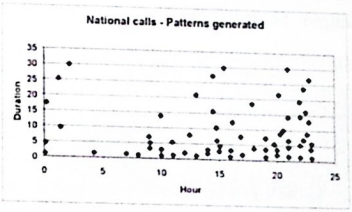


Fig. 3. Patterns generated after the training corresponding to national calls

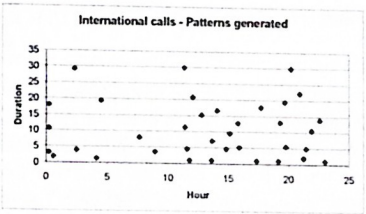


Fig. 4. Patterns after the training corresponding to international calls

The graph (Fig.2) shows the 144 patterns generated after the training of the neural network corresponding to local calls. At simple sight, it is easy to notice that there is a greater concentration of patterns in the time range between 8h and 20h and a duration of about 0–5 minutes. This denotes that most of the local calls made by this company's customers occur at these hours with the average durations indicated. The graph (Fig.3) shows the 64 patterns generated after the training of the national calls neural network. Here, also, a concentration of patterns can be seen, but this time more towards the time range of 15h to 22h with durations that vary between 0 and 7 minutes. It also shows

that there are practically no patterns generated for dawn, which may lead to conclude that most users of the company being analysed do not make any NAT calls during early hours. The graph (Fig. 4) shows the 36 patterns after the training of the international calls neural network. Here the distribution is a little more random, but the duration of the calls "chosen" as patterns tend to have a longer duration (between 7 and 10 minutes).

5.2. Construction of profiles and detection of changes in behaviour

In this section, the results presented were obtained after the construction (from the company records) of the profiles and the detection of the corresponding alarms for each of the two experiences made. The graphs show the CUP and UPH profiles at the moment an alarm was set off. On axis X, the 244 patterns (144 LOC, 64 NAT and 36 INT) are shown and on axis Y the distribution of frequencies of each of the patterns for the user being analysed at the moment the alarm was set off.

5.2.1. Experience 1 (Updating UPH with each call, high sensitivity with low Hellinger Threshold):

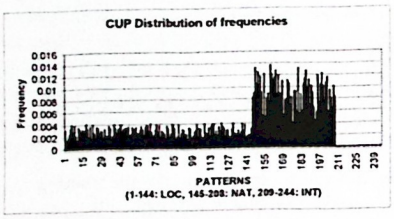


Fig. 5. User's CUP at the moment an alarm was set off

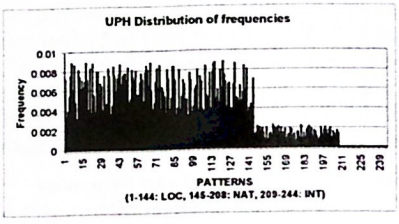


Fig. 6. User's UPH at the moment an alarm was set off

The graph (Fig. 5) shows a user's CUP at the moment an alarm was set off. It can be observed that the distribution of frequencies indicates a major tendency to make NAT calls (patterns 145 to 208). The graph (Fig. 6) shows the same user's UPH at the moment the alarm was set off. It can also be observed that the distribution of frequencies indicates a major tendency to make local calls (patterns 1 to 144). Hence, the difference between both distributions of frequencies defined by Hellinger distance (H) equals 0,30081. By analyzing the detail of this user's calls from dates previous to the triggering of the alarm to the day it was set off, there is evidence that the alarm responded to the user's making his first NAT call since his calls were processed. This means, his historic pattern of behavior did not make it evident that this user would make such a call. However, when these calls were made, the system detected the change and generated the corresponding alarm. These results also show that, having made the experience with such high sensitivity, one single different call can indicate a change in behavior that leads to an alarm. The total number of alarms that were set off after analyzing the 60 users was 88, out of which 33 correspond to different cases. This

is due to the fact that once an alarm for a user is set off, the following calls keep on setting off alarms till the UPH definitely adapts to the change in behavior. Most of the calls follow the pattern of the case in the graph in which a call that is different from the normal pattern of behavior is enough for the system to define the user as suspicious.

5.2.2. Experience 2 (Updating UPH once a day, moderate sensitivity with Hellinger threshold):

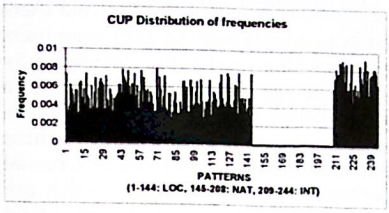


Fig. 7. User's CUP at the moment an alarm was set off

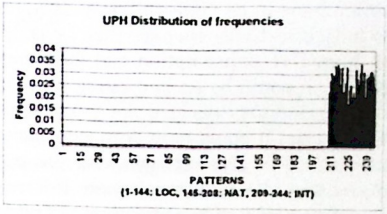


Fig. 8. User's UPH at the moment an alarm was set off

The graph (Fig. 7) shows a user's CUP at the moment an alarm was set off. It can be observed that the distribution of frequencies indicates a tendency to make local calls (patterns 1 to 144) and International calls (patterns 209 to 244). The graph (Fig. 8) shows the same user's UPH at the moment the alarm was set off. It can also be observed that the distribution of frequencies indicates a major tendency to make INT calls only (patterns 209 to 244). Therefore, the difference between both distributions of frequencies defined by Hellinger distance (H) equals 0,82815. By analysing the detail of this user's calls from dates previous to the triggering of the alarm to the day it was set off, there is evidence that the alarm responded to the user's making only international calls till the moment that he started making local calls. When the number of local calls modified the CUP in the way illustrated by the graph, the alarm was triggered. This is a particular case as surely this alarm is not an indicator of fraud if the user pays his invoice for international calls. But it is an indicator of a sensitive change of behaviour in the pattern of consumption, and that is exactly what this system searches. The total number of alarms that were set off after analysing the 60 users was 64, out of which 14 correspond to different cases. This is due to the fact that once an alarm for a user is set off, the following calls keep on setting off alarms till the UPH definitely adapts to the change in behaviour. This phenomenon is emphasised here because it is only after calls of the next day are processed that the UPH is updated. The majority of the calls follow the pattern of the case in the graph in which there must be several calls outside the pattern of behaviour for the system to find the user suspicious. This is much more satisfactory than what was obtained in experience 1 in which the high sensitivity showed users as suspicious simply for having made *one single* different call.

6. Conclusions

The results that were obtained were satisfactory in the sense that they were able to establish changes in the behavior of the users analyzed. Though the change in behavior does not necessarily imply fraudulent activity, it manages to restrict fraud analysts' investigation to this users' group. By using then other types of techniques [2], it is possible to obtain, with a high degree of certainty, a list of users who are using their mobile phone in an "unloyal" way. Besides, the experiences have helped to find users who have effectively changed their behavior, but in an inverse way, i.e. they were users with high INT consumption and then they started making local calls. Commercially speaking, it could be an interesting tip to evaluate this type of consumers, since, for a certain reason they decided not to use their mobile phones to make international calls any more and it could help draw conclusions and create new rateplans based on these situations. It is also proven, with the experiences carried out, that the differential analysis provides with much more information than the absolute analysis, which can only detect peaks of consumption and cannot describe the user in question. As a final conclusion, neural networks can be said to be excellent tools for the classification of calls and the construction of users' profiles as they represent their behavior in a faithful and efficient manner.

References

[1] ASPeCT, 1996. Definition of Fraud Detection Concepts, Deliverable D06. 47 pages.

[2] ASPeCT, 1997. Fraud Management tools: First Prototype, Deliverable D08. 31 pages.

[3] Beveridge M., 1996. Self Organizing Maps. http://www.dcs.napier.ac.uk/hci/martin/msc/ node6.html

[4] Burge P, Shawe-Taylor J. 1997a. Detecting Cellular Fraud Using Adaptive Prototypes, Department of Computer Science Royal Holloway, University of London.

[5] Burge P, Shawe-Taylor J. 1997b. Frameworks for Fraud Detection in Mobile Telecommunications Networks, Department of Computer Science Royal Holloway, University of London.

[6] Burge P, Shawe-Taylor J. 1997c. Fraud Detection and Management in Mobile Telecommunications networks, Department of Computer Science Royal Holloway, University of London. Vodafone, England. Siemens A. G.

[7] Hollmen J, 1996. Process Modeling using the Self-Organizing Map, Helsinki University of Technology Department of Computer Science.

[8] Kohonen, T. 2000. Self-Organizing Maps. Springer Series in Information Sciences

Selecting Pedagogical Protocols Using SOM

F. Salgueiro, Z. Cataldi, P. Britos, E. Sierra and R. García-Martínez
Intelligent Systems Laboratory. School of Engineering. University of Buenos Aires
Educational Informatics Laboratory. School of Engineering. University of Buenos Aires.
Software & Knowledge Engineering Center. Graduate School. Buenos Aires Institute of Technology
Intelligent Systems in Engineering Group. School of Engineering. University of Comahue
rgm@itba.edu.ar
(Paper received on August 11, 2006, accepted on September 26, 2006)

Abstract. During the first courses of the Computer Engineering undergraduate Program at the University of Buenos Aires, the number of human tutors in Programming Area is usually not enough: the students/tutors ratio is very high and there is a great heterogeneity in the acquired knowledge and background of students. The main idea behind this paper is to describe a system that could emulate a human tutor. Thus, the tutor will be able to provide student with a high level of flexibility for the selection of the most adequate tutorial type. This could be a feasible solution to the stated problem. But a tutorial system should not only emulate the human tutor but also it should be designed from an epistemological perspective concerning what teaching Programming really means. This is specially important in an Engineering course due to the required profile and identity of the future engineer. The stated solution implements a series of artificial neural networks to determine if there is a relationship between the given initial population of students learning preferences and the different tutoring types. A series of experiences were carried out in order to validate the current model.

1. Introduction

The main objective of the tutor module of an Intelligent Tutoring System is to present the new knowledge to the student in the best way possible. To achieve this, our research group [Salgueiro *et al.*, 2005; Costa *et al.*, 2005] has designed a series of sub modules and interfaces to avoid the normal overlap that usually appears in the modules of an Intelligent Tutoring System. In the tutor module, the main sub module contains the pedagogical protocols, which is made up of two basic components: the profile analyzer and the database of pedagogical protocols available in the system. The system has a database of pedagogical protocols. Its use will be subordinated to the availability of the contents in the knowledge module, although the lesson always can be generated for some of the available protocols. In order to collect data about the way in which each student learns, lists of learning styles will be used as well as tools for data collection. It has been determined the validity and trustworthiness of this instrument through its application by various researchers from the date of its creation [Felder, 1988; Figueroa, 2004] up to now. Starting from data provided by each student, his or her learning style will be determined. Afterwards, in a second step, the learning style will be linked to the

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 205-214

pedagogical protocol. The Felder list [Felder 1988] is a validated tool that allows obtaining solid data from students. After giving a questionnaire to the students, we will try to get data records on different sets by using the tools provided by Artificial Intelligence (AI), such as Neural Networks (NN) in order to obtain a relationship between the preferences of the students and pedagogical protocols. From a statistically significant sample of students for which the lists of complete learning styles had been taken, will try to see if the learning styles can be grouped according to the education techniques or pedagogical protocols. This will allow correlating the preference of the student with the most suitable pedagogical protocol in the system. As the selection of the pedagogical protocol is one of the elements to determine, it is desired to group the students in families with common characteristics.

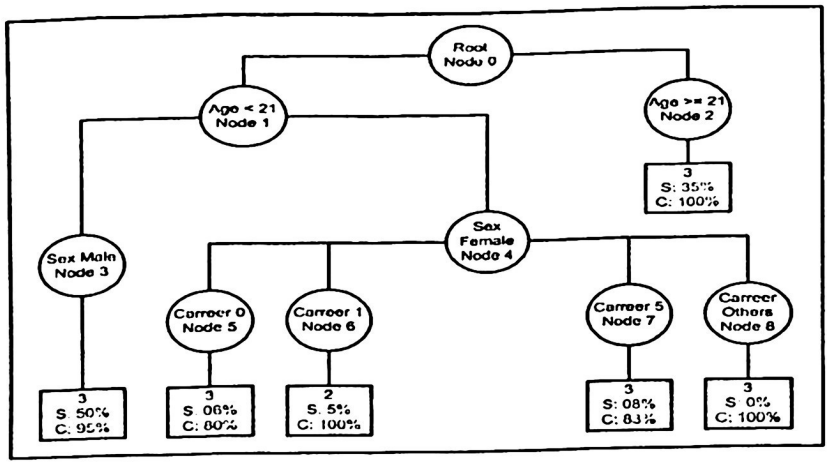


Fig. 1. Tree generated by C4.5 algorithm using SOM output as C4.5 Input.

This can be achieved using the Self Organizing Maps (SOM) neural networks (also known with the name Kohonen [2001] maps) that make a "determined clustering" or grouping action according to common characteristic of the original set of individuals. Once obtained the resulting groups of SOM network an induction algorithm will be used to find the rules that characterize each one of these groups. In this case the algorithms to be used will belong to the family of Top-Down Induction Trees (TDIT) algorithms. Although several algorithms exist that make these functions, a very complete one is Quinlan's C4.5 [Quinlan, 1993], an extension of algorithm ID3 (Induction Decision Trees) also proposed by Quinlan [Quinlan, 1987]. Its objective is to generate a decision tree and the inference rules that characterize this tree. In this particular case, the C4.5 will take as input the data of the students already clustered by SOM and the output will be the rules describing each cluster.

Once obtaining the smallest amount of rules by pruning the tree to avoid overfitting, we move to another stage of the analysis in which, by means of an inference process,

we found the relation between the SOM clusters and the pedagogical protocols available. In order to carry out the inference, additional data concerning the performance of students with different protocols of education in the courses under study were used. In Figure 2 the scheme of the solution can be seen: it represents the process of selection in a global form, where we start from a student population for which we have their preferences concerning learning styles through the lists of Felder. We form groups of students by using SOM. A table is generated using the previously classified students, using all the attributes that describe them and the cluster predicted by SOM. Then C4.5 algorithm is used to generate the rules that best describe each cluster, relating a particular cluster not only with all its attributes, as in the table of classified students, but also with a set of rules.

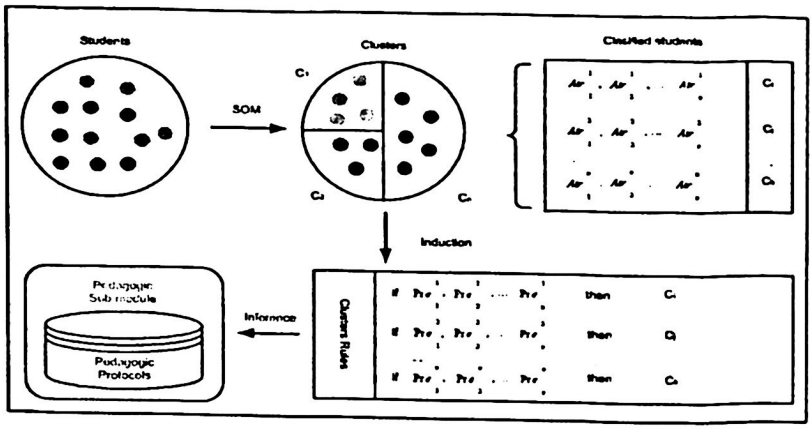


Fig. 2. Basic scheme of the solution.

2. Inference of the pedagogic protocol

In this stage we try to relate the groups generated by SOM to the pedagogical protocols by training a Backpropagation type neural network. In order to find the relationship between the learning style and the pedagogical protocol that best fits each group, the basic protocols described by Perkins [Perkins, 1995] in Theory One were used: [a] The *didactic or skillful instruction*: It satisfies a need that arises within the framework of the instruction in order to expand the repertoire of student knowledge [b] *The training*: It satisfies the need to make sure the student will have an effective practice, [c] *Socratic education*: This type of instruction is applied to provide educational aid to the student to include/understand certain concepts by himself and to give him the opportunity to investigate and learn how to do it, all by himself. Therefore the research is oriented towards the search of the relationship between the predilection of students concerning learning style and the pedagogical protocols used by the human tutors (professors). In

order to accomplish this, the grades of the partial evaluations are used. Two courses (A and B) will be taken belonging to the Basic area of Programming. The only fundamental difference between both of them was centered in the form of education, that is to say, in the pedagogical protocol used in the classes. From this frame of reference, two courses were evaluated according to the control variables raised by García [1995]. The variables raised for the reference courses are the following ones: [a] Similar contents of the courses, [b] Similar schedules, [c] Similar bibliography used for references, [c] Random entrance of the students, without preference defined to some course, [d] Similar previous formation of the assistants and instructors in charge of practical works, [e] Similar didactic tools an [f] Way in which the class is dictated, where each one of the tutors presents the classes based on the pedagogical protocol that turns out more natural to carry out to him. The possible options are defined in Theory One and that are analyzed in this investigation, independently of the needs or preferences of individualized students. From the analysis made by García, it is observed for this study where the only variable that changes is the one denominated "way of class dictation", that is to say, the pedagogical protocol used for each course. In order to carry out the inference, the following hypothesis will be considered: It is possible to relate the learning styles to the pedagogical protocols. Two more particular hypotheses arise from this main one: (a) The composition of styles of learning (needs and preferences of students) of each student determine the style of education (or pedagogical protocol) (b) Those students for whom the education style does not agree with their preference, show difficulties in the approval of the taught subjects. From the second hypothesis it is given off that for the approved students, the protocol preferred by most of them will have to be the one that agrees with the used one in class by the tutor, whereas for the failed ones, the protocol must be inverted for most of them. In order to validate this affirmation a network of Backpropagation type was trained with the following characteristics: [1] the approved students of the course with professor who dictates in Socratic style and the most of the failed ones of the course with professor who dictates in skillful way and the network is trained considering as output the Socratic protocol. [2] the approved students of the course with professor who dictates in skillful style plus the failed ones of the course with professor who dictates in Socratic way and the network is trained considering the output exit as skillful protocol. In order to suppress the "data noise" the training is carried out in the previously indicated way due to the fact that the groups that are outside the analysis contribute to increase the data noise (those students that approved with any protocol which will be considered "indifferent" and those that failed by lack of study or other reasons) and hope that the error of the tool is minor than the percentage of elements that are outside the analysis. Therefore, each generated cluster will be analyzed in the following way:

- approved students
  - Correct protocol → majority class
  - Indifferent → minority class
- failed students
  - Inverted protocol → majority class
  - Lack of study → minority class

Now we look to relate the forms of education and the learning styles. In Figure 3 it is observed, taking as base an example on where two pedagogical protocols exist and two preferences in the set of students, that the students whose preference agrees with the

form or style of education do not have problems to approve. Two subgroups (in red) of students exist whose preference does not agree with the education form and are those that fail, since they are "bad located".

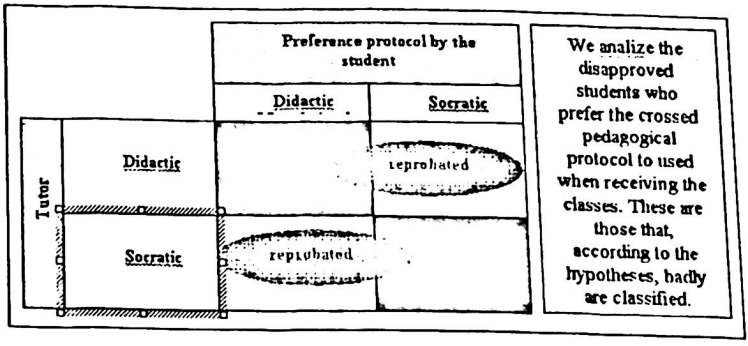


Fig. 3. Inference general scheme

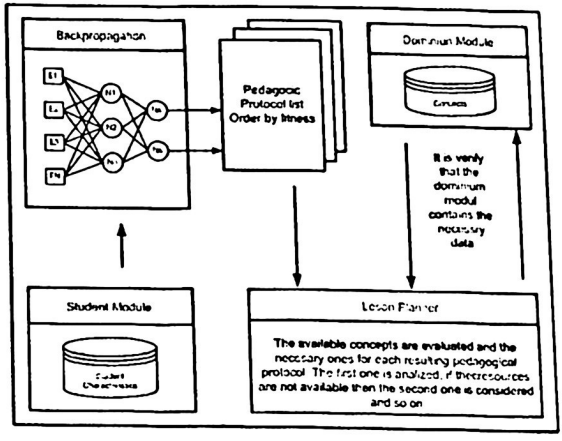


Fig. 4. Modified solution structure

Following the hypothesis: failed students who do not belong to the main cluster predicted by SOM must have a different preference concerning a pedagogical protocol (inverted in this case) from the one the professor used when they attended the classes. On the other hand, Figure 3 displays the idea of the hypothesis, where the failed students, who do not belong to the main cluster, must have a preference of pedagogical protocol different from the one received in the classes. Therefore, if the data provided by the educational system is analyzed in both courses with respect to the

categorizations made by system SOM, it is possible to be obtain which is the percentage of students who would be badly located in the courses, and this is demonstrated through the bad results reprobated obtained in the evaluations. So that the obtained results are satisfactory, the Backpropagation neural network must have a classification error smaller than the percentage of elements that were left outside of the analysis. This way this tool will be useful for the classification of the preferences of education (pedagogical protocol) of students according to their styles of learning. This way, a submodule gives a ranking of best suitable pedagogical protocol, in descendent order with respect to the preference for the selected student. Soon, the only thing that is required is to cross all the pedagogical protocols including in the system. The basic scheme of the solution can be seen in Figure 4 in terms of the student groups and the selection of possible protocols, where the Backpropagation network provides a ranking of aptitude of the pedagogical protocols available in the system.

3. Experimental results

The experimental results will allow to validate the proposed solution and will establish the steps in order to repeat the experiences with another group of students or to apply the method in other knowledge domains different from the Basic Programming. The fundamental steps for the experimental design are described in Table 1 where it is started with the taking of data of the students (to lists of learning styles) and it is used them like entrance for the training of a neural network of SOM type to generate the different groups. Soon the rules identify what describes these groups by means of the C4.5 algorithm.

Table 1. Steps for the experimental design

Step	Input	Action	Output
1	Data recollection from students	Use Felder tool on students	Result of the Felder tool.
2	Felder tool result	SOM Training	Students Clusters
3	Cluster + Felder tool results.	Use C4.5 algorithm	Rules describing each generated cluster and the corresponding decision tree.
4	Academic performance	Academic data grid	Academic grid
5	Result of the Felder tool + Academic grid + Clusters	Analysis of the cluster and determination of reprobated students.	Reprobated Student List for each cluster.
6	Result of the Felder tool.	Backpropagation training	Determination of the training error and the data out of analysis. Find the relation between learning style and pedagogic protocol.

4. Validation of the population size

In order to determine the minimum number of elements in the sample we use Hernández Sampieri [Hernández Sampieri, 2001] for the calculation. An initial of 800

student's population has S2 variance of the sample of n student that can determine in terms of the probability p where:

$$V = 0.03 \quad (5.1)$$

$$V^2 = (0.03)^2 = 0.0009 \quad (5.2)$$

The number of samples without any adjustments will be:

$$n = (S^2 / ^2) = 0.09\ 0.0009 = 100 \text{ students.} \quad (5.3)$$

Adjusting in order of the real N population:

$$n = (n' (1 - n' N)) = 100 (1 - 100\ 800) = 89 \text{ Students.} \quad (5.4)$$

The generalization error is below 3%, with which it is possible to say that the sample size is representative for all the students of the courses. Now we are ready to train the SOM network using the data collected from the Felder tool. Most of the parameters of SOM network arise through an iterative process, where the network trains and the results are analyzed. If the results are satisfactory (that is to say, the training error is then sufficiently small), the parameters are modified slightly to try to improve them still more. If the results are little satisfactory they are compared with previous set and they are modified in a higher value.

Table 2. Parameters used for SOM with which the data of the students were classified.

Parameter	Value
Observations	121
Variables	47
Artificial Neurons	10
Cicles	1000
Aleatority	Yes
Learning Parameter	
Initial	0.9
Final	0.1
Decay Function	Exp
Gaussian boundary parameter	
Initial	99,0%
Final	01,0%
Decay Function	Exp

It is possible to indicate that for obtaining the final values for training the neural network they have been proven more than one hundred combinations, obtaining the best results with the list of parameters that is observed on Table 2.

The amount of clusters: If the amount of clusters is very high, it may occur that it does not exists a correlation between so many pedagogical protocols and clusters, since it is started from the hypothesis that 3 pedagogical protocols exist (the proposed by Theory One). The number of clusters that is expected to get will be annotated between two and three. Summary of the results that the training of the SOM networks gives is in Table 3, where the elements of each cluster generated are totalized and the respective percentage are indicated.

Table 3. Summary of resulting elements after applying SOM to the input data.

	Cluster 1	Cluster 2
Data with all the attributes	6 (5.00%)	114 (95.00%)

The result is within the awaited amount of clusters and therefore the experimental data, they agree in the amount of clusters generated. As all the data are categorical, the

generated rules will be equalities and it will not be found any range for them (for example: the continuous data). In order to find the attributes with greater gain of information, it is required to use the first N passages of the C4.5 Algorithm. In this case, the first nine were taken and the rules appear in Table 4.

Table 4. Resulting rules to cross the tree generated by the C4.5 Algorithm

Rule	Antecedent	Consequent
Rule 1	If "Normally they consider me: Extrovert"	Then Cluster 2
Rule 2	If "Normally they don't consider me Reserved neither Extroverted"	Then Cluster 1
Rule 3	If "I Remember easily: Something that I have thought much"	Then Cluster 2
Rule 4	If "I don't remember easily something than I have thought much or something that I did"	Then Cluster 1
Rule 5	If "I learn: To a normal rate, methodically. If I make an effort, it profit".	Then Cluster 2
Rule 6	If "I do not learn to a normal rate, not methodically neither disordered"	Then Cluster 1
Rule 7	If "When I think about which I did yesterday, most of the times I think about: Images"	Then Cluster 2
Rule 8	If "When I think about which I did yesterday, most of the times I think about: Words"	Then Cluster 2
Rule 9	If "When I think about which I did yesterday, most of the times I don't think about words neither images"	Then Cluster 1

Oates [Oates *et al.*, 1997] among others has analyzed several algorithms of "pruning" to adjust the size of the rules generated from a great number of observations. Oates has found that as the size of the initial observations is increased, the size of the rules increases in linear form. This increase in the amount of rules antecedents does not significantly increase the precision in the classification of the rules. Continuing this way we get a result, for this case in individual, as proposed in the works of Quinlan [Quinlan, 1987] and Oates [Oates *et al.*, 1997] previously mentioned, offered the additional advantage when using the second tree (with less levels and minor amount of nodes), the Intelligent Tutorial System requires minor amount of information to select the pedagogical protocol of the student and with easier access information (it is simpler to know the answers of some key questions in the list that the answers to all the questionnaire). Training this way it is managed to suppress the "noise" that contributes the groups that are outside the analysis. In Table 5 the results of the students discriminated by courses can be seen, counting total students, students failed classified as belonging to the cluster in opposition to the one of the majority and the percentage that relates the failed and approved students that in addition are bad classified. For this experience the network of the Backpropagation type trained and a ranking (scale) of pedagogical protocols most adapted for a particular situation was obtained, in order to give flexibility to the module that stores the contents.

Table 5. Summary of percentage obtained for the analysis of students, discriminated by courses.

Observed Characteristic	Course A	Course B
Total of Students (For this study)	47	53
Students who reprobated the partial evaluation and were in a course with different pedagogical protocol	30	0
Students who approved the partial evaluation were in a course with different pedagogical protocol (inverted)	10	33
Approved students (no mattering about the protocol)	7	20
Reprobated students respect to the approved ones, within the subgroup of badly classified	75%	0%

For the training of the Backpropagation network 67% of the data (qualifications) were used randomly whereas 33% of the remaining data were used to validate the generated model. After more than 100 training of 1000 cycles each one, where it was carried out in order to diminish the error in the resulting network, it was reached the conclusion that the optimal values for the parameters of the network are those that are seen on Table 6.

Table 6. Neural Net Training Results.

Characteristic	Value
% Error (Training group)	3.75%
% Error (Validation group)	2.00%
Network characteristics	
Input neuron	13
First hidden layer neurons	20
Second hidden layer neurons	20
Output neurons	2

This training is valid since the error of the tool (3.75% for the set of training and 2.00 % for the validation set) is minor than the error of the elements that were outside the analysis, which represents the students who did not approve because lack of study, although the pedagogical protocol agreed with the preference of the student (who is 25%). Therefore it is possible to conclude that: [a] course B is related to cluster 1: since the errors induced by elements of cluster 2 within the course are in a 75% or in other words, the network classifies to 75% of the students failed in the course and [b] course A is related to cluster 2: since another possible allocation in this case does not exist and in addition the percentage error of classification and reprobation is of 0%. The obtained results agree with the affirmations of Perkins, where the Backpropagation network predicts that most of the failed students must have received classes using another pedagogical protocol. Socratic protocol is related with Cluster 2 and Magistral protocol is related with Cluster 1. This way the same turn out of the inferential step is obtained. In order to incorporate the experimental results to the design of the tutorial module of Intelligent Tutoring System (ITS), certain conclusions can be established. One concludes, that controls a module of the tutor able to categorize to the students according to its characteristics, within some of the pedagogical protocols available in the system. for the case in study, controlled data of 2 pedagogical protocols (Magistral and Socratic) and in this case is possible to be categorized automatically to the students within each one of them, according to its preferences to improve the results of a pedagogical session.

5. Conclusions

When validating the model against the real data. as much for the data triangulation as the training of the neural networks that support the model. it was found that the data adapt very satisfactorily to the test conditions, becoming thus, not only a theoretical tool being worth to guide the students in the learning process, but also in a validated instrument. In practice, that allows implantations of an Intelligent Tutorial System able to generate measurable and useful satisfactory results in real environments. It is

fulfilled then the primary objective of this work which is to provide an additional tool for the human tutors, who can relegate some of their tasks that, either by lack of time or resources, cannot fulfill in a satisfactory way the student request, whereas it provides a secondary support for the students who try to complement their knowledge or to regulate its own rate of learning. Then, it is provided to the field of the Intelligent Tutorial Systems a new tool, to facilitate the selection of the suitable pedagogical protocol, resulting this in a gain, not only for the performance of the STI itself, but also for the student, who is the fundamental human component that makes the system useful and offers identity to them. Thus it was tried to make a contribution to improve the academic performance of different students and therefore their quality of life.

References

Costa, G.; Salgueiro, F., Cataldi, Z., García-Martínez, R. y Lage, F. (2005). Intelligent Systems for student modelling. Proc. GCETE'2005, Global Congress on Engineering and Technology Education CD. march 13-15. Brazil.

Felder R. & Silverman L. (1988). Learning Styles and Teaching Styles in Engineering Education. Engr. Education, Volumen 78, número 7, p. 674-681.

Figueroa, N.; Lage, F.; Cataldi, Z.; Denazis, J. (2003). Evaluation of the experiences for improvement of the process of learning in initial subjet of the computer science careeer. Proc. Int. Conf. on Eng. and Computer Educ. ICECE 2003, March16-19. San Paulo. Brazil.

Figueroa, N.; Cataldi Z.; Salgueiro F. A.; Costa G.; Méndez P.; Lage F. J. (2004). The styles of learning and the university desertion in Computer science Engineering. CACIC 2004. Arg.

García, M. (1995). El paradigma de Pensamiento del profesor. Editorial CEAC, Barcelona.

Hernandez Sampieri, C. (2001). Metodología de la investigación. Mc Graw Hil. México.

Kohonen, T. (2001). Self-Organizing Maps, third edition. Springer.

Quinlan, J. (1987). Simplifying Decision Trees. Simplifying Decision Trees. International Journal of Man-Machine Studies 27(3): 221-234.

Quinlan, J. (1993). C4.5: Programs for Machine Learning. Morgan Kaufmann.

Salgueiro, F., Costa, G., Cataldi, Z., García-Martinez, R. y Lage, F. (2005). Intelligent Systems for tutor modelling. Proc. Global Congress on Eng. and Tech. Education. march 13-15

Oates, T. (1997). The Effects of Training Set Size on Decision Tree Complexity. Proc. 14th International Conference on Machine Learning.

Perkins, D. (1995). Smart schools. The Free Press. A division of Simon & Schuster, Inc.

Support Vector Machines

Drowsy Driver Detection Using Wavelets and Support Vector Machines

Rodolfo Ibarra-Orozco[1], Miguel González-Mendoza[2,1], Neil Hernández-Gress[1], Ariel García-Gamboa[1], and Jaime Mora-Vargas[1]

[1] Department of Computer Science. ITESM, Campus Estado de México
Carretera Lago de Guadalupe Km. 3.5, Atizapán de Zaragoza, Estado de México
{rodolfo.ibarra, ariel.garcia, mgonza, ngress, jmora}@itesm.mx

[2] LAAS CNRS.
Avenue du Colonel Roche. 31077 Toulouse Cedex 4 France.
mgonza@laas.fr

(Paper received on September 05, 2006, accepted on September 27, 2006)

Abstract. Driving behavior emerges from the intricate and complex interactions between the driver, the vehicle and the environment. The purpose of a behavioral model is to explain and to predict the human behavior. For the construction of this kind of models, mathematical and Artificial Intelligence techniques have been tested. Some constraints of the system are 1) the supervision and monitoring of the driver-vehicle-environment system is a complex problem due to particularities such as stochastic, personal and dynamic. 2) The solution must be based on a non-intrusive system and 3) able to diagnose and predict hypovigilance from peripheral on-board sensors. The proposed diagnosis system is based on intelligent signal processing algorithms and real time methodologies as the Wavelets analysis and the Support Vector Machines learning approach for density estimation. In this paper, we search to accomplish the requirement of a faster and robust algorithm, able to work in real time. Results are validated using driver's data in real conditions. The framework of this study is the European research project SENSATION.

1. Introduction

Human factors, as excessive fatigue, extended inattention or stress, have been the key cause of many industrial accidents. Most nuclear accidents (Chernobyl, Three-mile Island and several US islands), the Challenger explosion, 40% of road accidents and, according with the NASA, approximately 21% of the aviation incidents are fatigue related, [1].

Driving is a complex behavior influenced by a wide range of factors in space and time. Factors include goals, distraction, errors, expectancies, workload, attention, traffic, vehicle safety features, automation level, fatigue, memory, capabilities, training and experience. Driving behavior is conceptually considered as a complex system in which the environment, driver and vehicle are influencing factors. An emerging driving behavior could not necessary be linearly predicted. However, a context-aware system could assist the driver in augmenting the probability of undertaking a safe behavior. To detect drowsiness in a human being, physiological measures, such as brain activity, can

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 217-225

be used. The main disadvantage of this kind of measures -despite their accuracy is the contact requirement with the driver. For automotive systems, unobtrusive techniques must be used. Further, the ideal system will detect the precursors to drowsiness at least several minutes before onset, giving the driver time to rest or take another action. In order to monitor both the driver and the road environment to detect hypovigilance in real-time, it is necessary to integrate multiple parameters. Information on the road environment, personalized driver characteristics and advanced detection techniques must be fused to create a robust system.

This paper is organized as follows: Section two gives an introduction to Wavelets, Support Vector Machines for density estimation and CUSUM. Section three shows the drowsy driver detection proposed methodology. In section four, obtained results on driver's data in real conditions are presented. Finally, section five gives some conclusions.

2. Background

2.1. Continuous Wavelets

Wavelets, [3], allow analyzing in time and frequency the behavior of a signal. They are able to analyze non-stationary signals, in complement of the Fourier analysis that loses the information of temporal localization. The Wavelets analysis is based on a convolution of a Wavelet mother ψ which one dilates (s) and translates (τ), see equation 1, to have an approximation and a decomposition of the original signal.

$$\psi_s^{\tau}(t) = \frac{1}{\sqrt{s}} \psi\left(\frac{t-\tau}{s}\right) \tag{1}$$

Wavelets are known to be efficient in representing piecewise smooth functions. Away from singularities, the inner product between a wavelet (with a number of zero moments) and a smooth function will be either zero or very small. At singular points, a finite number of wavelets concentrated around the discontinuity lead to non-zero inner products. This is in contrast with Fourier series where discontinuities lead to many larger coefficients. This ability of wavelets expansions to capture both smooth and singular parts of a signal has been used in many applications, including denoising and compression.

2.2. DiscreteWavelets

Calculating wavelet coefficients at every possible scale is a fair amount of work, and it generates an awful lot of data. It turns out, rather remarkably, that if we choose scales and positions based on powers of two - so-called dyadic scales and positions - then this analysis is much more efficient and just as accurate. We obtain such an analysis from

the discrete wavelet transform (DWT). An efficient way to implement this scheme using filters was developed in 1988 by Mallat, [3]. The Mallat algorithm is, in fact, a classical scheme known, in the signal processing community, as a two-channel subband coder. This very practical filtering algorithm yields a fast wavelet transform a box into which a signal passes, and out of which wavelet coefficients quickly emerge. The filter bank is read from left to right. The input is a vector; perhaps arising as samples of a continuous signal/function. This is a two-channel filter bank. In brief, its effect is to 1) Separate the input into frequency bands: filter and downsample and, 2) Reassemble: upsample and filter. The filters could each be applied several times. One applies the H_0 lowpass (average) filter repeatedly at higher and higher levels, each branch followed by one H_1 highpass (difference) filter. See Fig. 1.

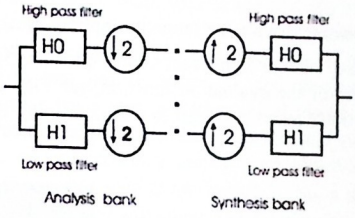


Fig. 1. Wavelet Filter Bank.

The left half of the filter bank is called the analysis bank, and its effect is to compute the wavelet coefficients of the signal vector. The right half of the filter bank is called the synthesis bank: it reconstructs the signal vector from the wavelet coefficients.

2.3. Support Vector Machines for Density Estimation

For Support Vector Machines (SVMs) in density estimation, [4], only one class have to be separated of the origin with an optimal hyperplane, Fig. 2. In order to build a SVM in the density estimation problem, with a nonlinear decision function, in a nonlinearly-separable case, one has to solve the following dual quadratic optimization (QP) problem:

$$\begin{aligned} &\underset{\alpha}{Maximize}\ L_D(\alpha) = -\frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j k(x_i, x_j) \\ &subject\ to:\ \sum_{i=1}^{l}\alpha_i = l, \\ &0 \le \alpha_i \le \frac{1}{\nu} \quad i = 1, \ldots, l \end{aligned} \tag{2}$$

where k(,) is the kernel function, which makes a projection of the original space into a space (feature space) of a higher dimension. α are the Lagrange Multipliers introduced to transform the primal quadratic problem with linear constraints into the dual problem showed above. The parameter $\nu \in (0, 1)$ establishes an upper limit of the fraction of vectors remaining out of the learned distribution (outliers) and a lower limit of the fraction of support vectors. The evaluation function is defined as:

$$f(x) = \sum_{i=1}^{l} \alpha_i k(x_i, x_j) - \rho \tag{3}$$

The examples x_i associated with Lagrange multipliers greater than zero are called support vectors, since they have a significant contribution in (5). Geometrically, these vectors are on the border of the evaluation function, see Fig. 2. Large scale learning tasks, with more than 10000 instances, lead to a long training time and a large kernel matrix.

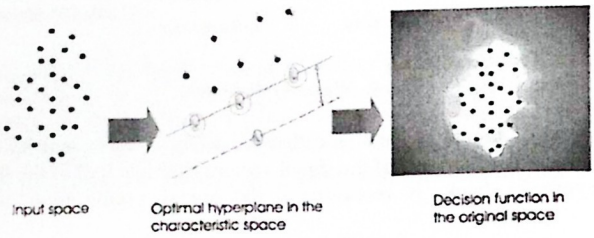


Fig. 2. Support Vector Machines for Density Estimation

Some solutions have been proposed in order to deal with this problem, [5], [6]. We have also proposed some heuristics, [13], [14], in order to reduce the training time.

2.4. Cumulative Sum (CUSUM)

CUSUM charts have shown to be efficient in detecting small shifts in the mean of a process. CUSUM works as follows: Let us collect k samples, each of size n, and compute the mean of each sample. Then the cumulative sum control chart is formed by plotting one of the following quantities:

$$S_m = \sum_{i=1}^{m} (x_i, \mu_0) \quad or \quad S'_m = \frac{1}{\sigma_x} \sum_{i=1}^{m} (\bar{x}_i, \mu_0) \tag{4}$$

Against, the sample number m, where is the estimate of the in-control mean and is the known (or estimated) standard deviation of the sample means.

3. Methodology

Practically, a system can be diagnosed by two principal ways depending on the available information. The first one, when a mathematical or structural model is known, diagnosis can rely on estimation, analytic redundancy and expert systems. In the second case the model is not known, so a behavioral model has to be constructed from available observations on the system. The use of statistics, pattern recognition, neural networks, and other methods is the principal way of parameterization of the model. The diagnosis system has to supervise and diagnose the driver-vehicle-environment system. As we have seen, a mathematical model of this system is quite impossible to be constructed due to the number of factors and variables to be taken into account. In this work, one constructs a model from observations: x in n, each observation represents the system state at the instant t. So, a number of observations belonging to normal driver behavior will create a cloud of points. The general approach is illustrated in Fig. 3. It features a Base Line Generation that computes the model parameters off-line. Then, for the Real Time Diagnosis these parameters are used to evaluate the driver's performance every 50 msec. The final diagnosis deals with the instantaneous one by statistical methods to provide a final diagnosis to the driver with a minimum of errors.

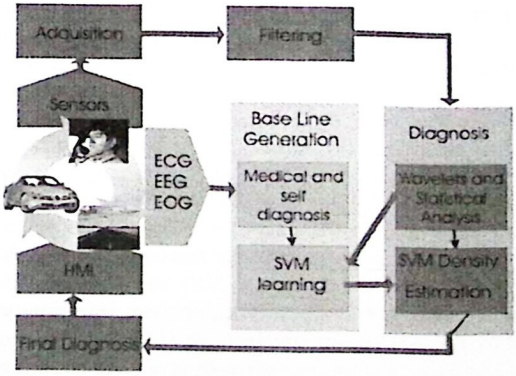


Fig. 3. General Diagnosis Methodology

A medical external expertise, based on EEG/EOG signals, is used as a target, in order to compare the resulting system. On-board measured variables are: lateral distance, steering-wheel angle and vehicle speed. The methodology is as follows:

1. **Singularities Detection.** On-board measured variables are used to compute a group of synthetic variables. By decomposing signals into elementary building blocks that are well localized both in time and frequency, the wavelet transform can characterize the local regularity of signals. This local regularity is measure from the wavelet transform modulus maxima. [10]. First, our algorithm detects singularities (points where there are a change of the dynamics' signal) as temporal references for compute statistical artificial variables: average, standard deviation and the time traversed between singularities.

2. **Data fusion.** The cloud of computed artificial variables is used to create a normal class by means of the estimation of the probability density estimation function (PDF). However, there are multiple constraints for this kind of diagnosis. To represent as accurately as possible the normal behavior we need an exhaustive number of observations in different states.

3. **Final Decision.** The final result is provided by CUSUM.

4. Experiments and Results

Field experiments are being carried-out in the SENSATION. All the experimentation campaigns were performed using the CopiTech demonstrator vehicle. Two important results of these experiments are, on one hand, the driving behavior is personalized and could change in time, because two drivers have different driving characteristics in normal and abnormal modes. On the other hand, the pertinent parameters depend on the road situation. In fact we have to distinguish between different environments.

The experimentation protocol can be seen as follows: Drivers drove twice in a day, morning and afternoon; doing 184 km at 90 km/h each time. They have been instrumented with electrodes with the aim of record their brain (EEG) and ocular (EOC) activity. At the same time the medical expert asked to the driver his state of tiredness. In addition, all signals coming from on-board sensors were recorded and the scene in the cockpit and the environment of the vehicle were filmed and recorded on videocassette. Ulterior analysis of EEG and EOG allowed to known the physiological state of the drivers. There are seven variables: three physic variables (Wheel Angle WA, Lateral Position LP and Vehicle Speed VS) and four artificial ones (standard deviation and frequency of WA and LP) calculated into intervals between two singularities detected by the wavelets analysis.

4.1. Singularities Detection

The local frequency of the oscillations can be measured from the points where the modulus of the wavelet transform is local maximum, both along the scale and the spatial variable. As an example, we show the lateral position's signal analysis.

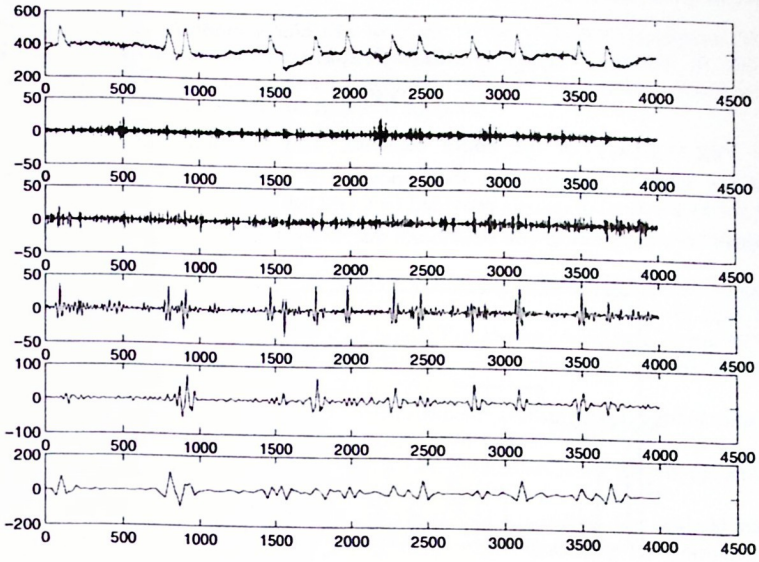


Fig. 4. Top: Analyzed signal. Bottom: Wavelet's detail coefficients at different levels.

To compute the wavelet transform, we use a complex Daubechies wavelet, due to the good capabilities of complex wavelets on detecting singularities, [11]. Both calculation of Wavelets and generation of artificial variables are done for the entire signal using fixed windows of 4046 samples, which make approximately 3.37 min for a sampling rate of 20 Hz. Different filter bank's output levels are shown in Fig. 4. Detail figures show the wavelets coefficients at different scales, higher level coefficient's modulus maxima shows where signal singularities take place.

4.2. SVM for density estimation and CUSUM for the final diagnosis

According to the process state, we can focus on problems in the driver's attention level. However, our interest is focused in the vigilant state. For one of the drivers, we have identified thanks to the medical expertise that the first 55 minutes belong to a vigilant (or normal) behavior and the rest to the hypovigilant one. For the learning process, we made use of the examples between 5 and 30 minutes of driving (30000 examples), obtaining 1399 support vectors for the learning parameter $\nu = 0.05$ and a Gaussian kernel with $\sigma = 0.35$. The results of the generalization procedure in the entire database are shown in figure 5. It shows at the top of the figure, the medical diagnosis

given in ten levels. The level ten belongs to a high level of hypovigilance. In the middle, we show the output of the learned SVM PDF, normalized by the function:

$$f(x) = \frac{\max(-f(x,0))}{reg} \tag{5}$$

where $reg = f(x_t)$ of the entire database. This function provides a classical Probability Density Function for the hypovigilant behavior. The lower part of the figure shows the final diagnosis provided by CUSUM. A numerical performance is not computed since the medical and behavioral diagnoses don't arrive at the same time.

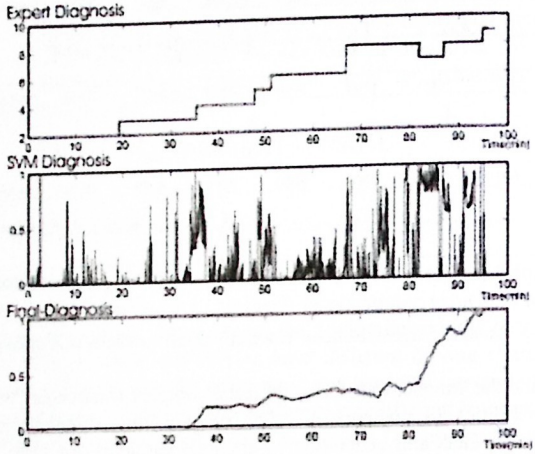


Fig. 3. High: External diagnosis expressed in ten levels. Middle: Wavelets-SVM punctual diagnosis. Low: Final diagnosis by CUSUM

5. Conclusions

This paper shows an Advanced Driver Assistance Systems (ADAS) focused in the diagnosis of a very complex problem: the driver's vigilance state. Several efforts are been or have been made by several teams all over the work to propose a reliable on-board and real time system. In this paper we have proposed a diagnosis system in order to perform the diagnosis of a vehicle driver in real time. The use of wavelets as an extraction methodology gives a more accurate description of the original physical variables and then the evolution over time. Wavelets joined to the Support Vector Machines shows its pertinence.

Acknowledgments. Authors would like to thank to SENSATION European Project (Advanced SENSor Development for AttenTION, Stress, Vigilance & Sleep / Wakefulness Monitoring). Contract number 507231. European Commission. http://www.sensationeu.org ., for its financial support.

References

[1] Mitler M et al. Catastrophes, sleep, and public policy: consensus report. Sep 1988;11(1):100-9.

[2] González-Mendoza, M. et al. Driver vigilance monitoring, a new approach. Intelligent Vehicle Symposium (IV'2002), Versailles (France), 17-21 June 2002.

[3] Stphane Mallat. A Wavelet Tour of Signal Processing. Academic Press, 1999.

[4] Vapniv, V. Statistical Learning Theory. Journal of the Association for Computing Machinery, 40:741–764. 1993.

[5] Osuna E. et al. "Support Vector Machines: Training and applications". In Massachusetts Institute of Technology, March 1997.

[6] Platt, John. Fast Training of Support Vector Machines using Sequential Minimal Optimization. Microsoft Research. MIT Press, 1998.

[7] Boser, B.E., Guyon I., Vapnik.V. A training algorithm for optimal margin classifiers Proceedings of the 5th Annual ACM Workshop on Computational learning Theory, 1992, pp. 144-152.

[8] Cristianini, N., Shawe, J. An Introduction to Support Vector Machines and other kernel based learning methods. Cambridge University Press. 2002.

[9] Vapnik, V. An Overview of Statistical Learning Theory. IEEE Transactions on Neural Networks, Vol. 10, no.5, September 1999.

[10] Mallat, S., Liang Hwang. Singularity Detection and Processing with Wavelets. IEEE Transactions on Information Theory, Vol. 38, no.2, march 1992.

[11]. Chung-Liang. Tu., Wen-Liang Hwang. Analysis of Singularities From Modulus Maxima of Complex Wavelets. IEEE Transactions on Information Theory, Vol. 51, no. 3, march 2005.

[12] Vapnik, V. and Chervonenkis, A. J. The necessary and sufficient conditions for consistency in the empirical risk minimization method Pattern Recognition and Image Analysis, vol. 1, no. 3, pp. 283-305, 1991.

[13] Garcia, A. Hernandez, N. Reducing the Training Time of Support Vector Machines by Barycentric Correction Procedure. WSEAS Transactions on Systems. ISSN 1109-2777. 2004

[14] Ibarra-Orozco, R. and Hernandez-Gress, N. Increasing the training speed of SVM, the Zoutendijk algorithm case ISSADS, Guadalajara. January. 2005.

[15] Vapnik, V. Computational Learning Theory. John Wiley & Sons. 1998.

[16] Garcia-Gamboa Ariel and González-Mendoza Miguel. A comparison of different initialization strategies to reduce the training Time of Support Vector Machines. International Conference on Artificial Neural Networks (ICANN), pp. 613-618. 2005.

Isolated Character Recognition by using Support Vector Machines and an Optimal Feature Vectors

Hermenegildo Cisneros, Ricardo Barrón and Benjamín Cruz
Centro de Investigación en Computación del IPN
Av. Juan de Dios Batis esquina Miguel Otón de Mendizábal
Colonia Nueva Industrial Vallejo
México, D. F. 07738, MÉXICO
E-mail: rbarron@cic.ipn.mx
(Paper received on September 06, 2006, accepted on September 27, 2006)

Abstract. In this work, we present a combination of techniques for the recognition of isolated hand-written characters. We describe a way to obtain the feature vector that allows us to describe an isolated character. Flusser invariant moments and other geometric measures are combined as object descriptors. We implement an off-line system. We use so-called Support Vector Machines [SVM] as the main classifiers.

1. Introduction

An object recognition system usually incorporate three main modules (Figure 1):

1. Data acquisition and preprocessing module.
2. Feature extraction module.
3. Decision (classification) module.

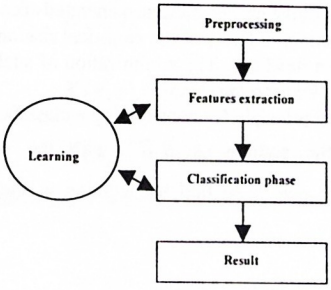


Figure 1. The three modules of an object recognition system.

After an image of an object (in this case the an image of the objects to recognize) is acquired, it is analyzed to get a compact description of the objects in the image. This compact representation model must fulfill certain desirable properties:

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 227-235

a) Unicity. Each object must have a unique representation.
b) Invariance. The description should experiment no changes under a group of image transformations: translations, rotations, changes of category, reflections, and the like.
c) Sensitivity. Capacity to discriminate equal objects.

These kind of descriptions are the so-called external representation schemes, which make use of the contour of the objects and its features, such that chain codes, Fourier descriptions, polygonal approaches. There are also the so called schemes of internal representation, that describe the region occupied by the object in a binary image, such area, perimeter, compactness factor, the moments, an so on. Some of these schemes can be found in [LP].

In this work we show how a careful selection of descriptors can help to recognize isolated written characters. In this work we have selected the well-known Flusser invariant moments, the elongation of the object, compactness factor, eccentricity and size. Flusser invariants have been chosen due to their capacity to recognize projected objects.

For classification we have used the multi-class support vector machines because their great capacity of classification, and because they are a safe alternative in regards to the most common methods: Bayesian, KNN, back propagation, among others.

2. Support vector machines

Most of pattern recognition methods are based on finding a classification function that minimizes the *empirical risk*, the error measure of the given patterns. The theory developed by Vapnik and Chervonenkis provides upper bounds of this *structural risk*, the error of misclassification of the unseen patterns generated according to an unknown but fixed probability distribution, in terms of the empirical risk and the VC-dimension of the family of classification functions. The minimization of such bounds follows the principle of *Structural Risk Minimization* [SRM].

Following the mentioned principle for constructing a classification function f(x) = sing (w·x+b) based on the patterns $x_i \in R^n$ with the corresponding labels $y_i \in \{-1,1\}, i = 1,....,l$ the goal is to find the solution w and b of the following optimization problem:

$$\text{Minimize} \quad \frac{1}{2} w + C\sum_{i=1}^{l} \xi_i \tag{1}$$

$$\text{Subject to} \quad y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad with \quad \xi_i \geq 0.$$

Here the quantity $\frac{1}{2} w$ the *margin*, is related to the VC dimension and is maximized. The parameter C can be regarded as a regularization parameter. To save some numeric

and implementation problems, the problem (1) is transformed using the technique of Lagrange multipliers into the following dual problem.

$$\begin{aligned} Max \quad & W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \\ s.t. \quad & \sum_{i=1}^{l} y_i \alpha_i = 0, \qquad 0 \le \alpha_i \le C, \qquad i = 1,\ldots,l \end{aligned} \tag{2}$$

An important characteristic of SVMs is that f can be expressed in terms of the training vectors x_i for which the solution α_i of (2) are positive, called the *support vectors* (*SV*s) [SV], by setting w and b as follows:

$$w = \sum_{\alpha_i \in SV}^{l} y_i \alpha_i (x_i \cdot x), \; b = \frac{1}{2}(w \cdot x_+ + x_-) \tag{3}$$

where x_+ and x_- are two SVs which belong to the positive and negative class, respectively. The SVs are interpreted as the relevant patterns in the classification process. Frequently, the number of SVs is small with respect at the number of training data; this gives a compact representation and efficient implementation of the classifier. Other important characteristic of SVM is that non-linearity can be introduced by replacing the inner product in (2) by a *kernel*

$$K(x_i \cdot x_j) = \phi(x_i) \cdot \phi(x_j),$$

where ϕ maps the patterns into a high (possibly infinity) dimensional inner product space. Some of the best known (and used) kernel functions are:

$$K(x,z) = \exp(-\gamma \, x - z) \tag{4}$$

$$K(x,z) = \exp(\gamma((x \cdot z) + 1))^d \tag{5}$$

$$K(x,z) = \tanh(\gamma(x \cdot z) - \theta) \tag{6}$$

which are know as *radial basis functions* (*RBFs*), *polynomial kernels* and *hyperbolic kernels* respectively.

2.1. Multi-class support vector machines

2.1.1. One-against-all (oaa) SVM

Consider an n-class problem. For oaa [OAA] SVM, we determine n direct decision functions that separate one class from the remaining classes. Let the ith decision

function, with the maximum margin that separates class *i* from the remaining classes, be

$$D_i(x) = w_i^T g(x) + b_i \tag{7}$$

where w_i is the *l*-dimensional vector, $g(x)$ is the mapping function that maps x into the *l*-dimensional feature space, and b_i is the bias term.

The hyper plane $D_i(x) = 0$ forms the optimal separating hyperplane, and if the classification problem is separable, the training data belonging to class *i* satisfy $D_i(x) \geq 1$ and those belonging to the remaining classes satisfy $D_i(x) \leq -1$. Especially, support vectors satisfy $y_i D_i(x) = 1$. If the problem is not separable, unbounded support vectors satisfy $y_i D_i(x) = 1$ and bounded support vector satisfy $y_i D_i(x) \leq 1$. Remaining training data satisfies $y_i D_i(x) \geq 1$.

In classification, if for the input vector x

$$D_i(x) > 0 \tag{8}$$

is satisfied for one *i*, them x is classified into class *i*. This is due to only the sign of the decision function is used. The decision is discrete.

If Eq. 8 is satisfied for plural is or if there is no *i* that satisfies Eq. 8, x is unclassifiable.

3. Implementation

3.1. Pre-processing

The pre-processing [LP] phase begins with the capture of the image to process; it is in BMP or JPG format. Figure 2 shows an example of an image with 5 isolated characters.

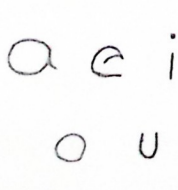


Figure 2. Original image

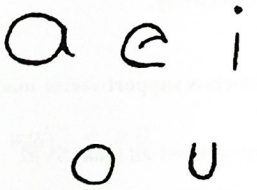


Figure 3. Threshold image.

Next a process to convert the image to grey levels is made, here the next phases are applied:

Thresholding. In this phase the image is taken from grey levels and a threshold is applied to it, so that the new image in two is obtained: 0 (black for the objects) and 1 (white for the background) (Figure 3).

Edge detection [DIPE]. Edges of each character are detected to find the limits among them (Canny's detector was used). Figure 4 shows the edges detected from figure 3.

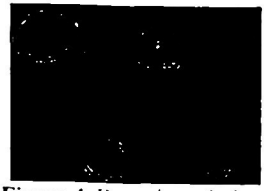


Figure 4. Detection of edges



Figure 5. Dilatation of characters

Dilation [DIPD]. The edges of each character are dilated. This because in some cases its lines are not completely closed or there are some isolated dots nearly at the main lines. By using dilatation these dots are joined at the lines. More formally, the dilatation of A by B note $A \oplus B$ and is defined by means of:

$A \oplus B = \{c \in E^n \mid c = a + b \text{ for some } a \in A \text{ y } b \in B\}$. As the addition is commutative, the dilatation also is it $A \oplus B = B \oplus A$.

It is possible to demonstrate that the following definitions of the dilatation are equivalent:

$$A \oplus B = \left\{ x \mid \left(\hat{B} \right) x \cap A \neq 0 \right\} \cup b \in BA_b$$

Actually the A and B sets are not symmetrical. The first element of the dilatation, A, is associated with the image being processed, the second one is named as structural element. The result of applying B on A produces the dilatation of A, $A \oplus B$ (Figure 5).

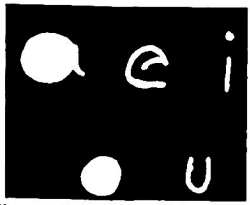


Figure 6. Filling of characters

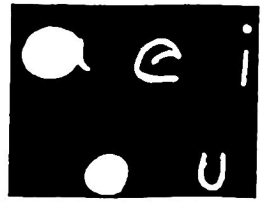


Figure 7. Labeled of characters.

Filling. A filling of pixels is made at the characters, to give them greater form. Figure 6 shows the filled characters of Figure 5.

Labeling [LP]. Labelling of characters is made to introduce them in the classifier, by using the labelled of connected components. Later we get the features vector of each object. Figure 7 shows the labelled characters from Figure 6.

3.2. Distinction of characters

Before continuing with the following stage of the system, some aspects are mentioned that we considered can help us to make a greater distinction of the characters. For example, we can observe that some high characters as the capital letters and some small letters exist (b, d, h, g, f, j, k, l, q, p, t...), as well as we observed low characters and in this category we found to the remaining small letters.

Also the ambiguous characters exist that are those in which the version in capital letter is morphologically equal to the small letter (K/k, P/p, Y/y, I/i I/l), and other that are not ambiguous but their height are very similar (P/f, h/k, j/g) among others.

In this sense, our system makes the classification on the basis of some of the aspects mentioned before. For example, if the character to recognize is greater to certain height, then it belongs to the category of high characters, otherwise to the class of low characters.

3.3. Feature vector

Since we have mentioned early, to get the feature vector of the objects we have used a combination of techniques. This with the idea to find a set of optimal features that allows us to recognize isolated handwritten characters.

In this work we show a selection of descriptors using the Flusser invariant moments, due to the capacity that they have to recognize projected objects (we considered that they can be very useful since many handwritten characters have this characteristic of projection) in addition to the elongation of the object, compactness factor, eccentricity and size.

Flusser invariant moments

Affine invariant moments [IMF] are derived by means of the theory of algebraic invariants, are invariant under general affine transformations.

$$\begin{aligned} u &= a_0 + a_1 x + a_2 y \\ v &= b_0 + b_1 x + b_2 y \end{aligned} \tag{9}$$

The affine transformation (1) can be discomposed in six parameters of transformation:

$$\begin{array}{llllll} u = x + a & u = x & u = \omega \cdot x & u = \delta \cdot x & u = x + t \cdot x & u = x \\ v = y & v = y + \beta & v = \omega \cdot y & v = y & v = y & v = t' \cdot x + y. \end{array}$$

Any function F of moments that are invariant under these six transformations will be invariant under the general affine transformation. Next, we show the parameters used for the obtaining of the feature vector:

$I_1 = (u_{20}*u_{02} - u^2_{11})/u^4_{00}$

$I_2 = (u^2_{30}*u^2_{03} - 6*u_{30}*u_{21}*u_{12}*u_{03} + 4*u_{30}*u^3_{12} + 4*u^3_{21}*u_{03} - 3*u^2_{21}*u^2_{12})/u^{10}_{00}$

$I_3 = (u_{20}*(u_{21}*u_{03} - u^2_{12}) - u_{11}*(u_{30}*u_{03} - u_{21}*u_{12}) + u_{02}*(u_{30}*u_{12} - u^2_{21}))/u^7_{00}$

Compactness factor

$$F = \frac{4*\pi*A}{P^2}, \quad \pi = 3.1416.$$

P = perimeter.
A = Area.

Elongation

$$E = \frac{P^2}{A}.$$

Size

$$Size = \frac{P}{A}.$$

Eccentricity

$$e = \sqrt{1 - \frac{b^2}{a^2}}.$$

b: smaller semi axis
a: greater semi axis.

4. Experimental results

The tests were made with the vowels and the numbers of 0-9.The letters were written by a person. For the case of the vowels, 50 letters were used for each class, that is, a total of 250 samples. For each class we selected in random way 50% of data for training and 50% for the tests.

For the case of the numbers, 50 digits were used by each class, that is, a total of 500 samples. As it happened with the vowels, we selected at random 50% of data for training and 50% for the tests.

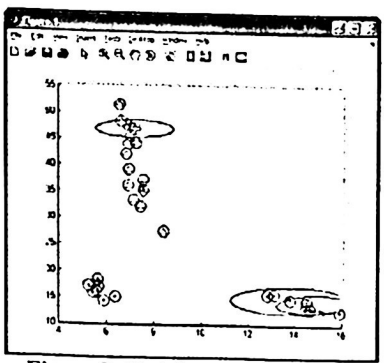


Figure 8. Training vowels with oaa SVM.

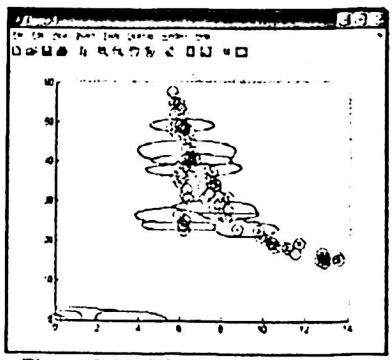


Figure 9. Training numbers with oaa SVM.

Our system uses oaa (one against all) as multi-class SVM classifier. We use the RBF kernel for all classification nodes with the implementation of optimized minimum sequence [SMO] to train and a constant of 10. The programming was made in Matlab using a Toolbox of Support Vector Machines [Stprtool].

To compare the results we trained a neural network with the RPROP algorithm (with standard parameters values) and the Euclidean classifier.

In Figures 8 and 9 we respectively show the training of data of the vowels and the digits with multi-class oaa SVM.

The following tables summary the results obtained for each classifier.

Table 1. Vowel recognition.

Vowels	SVM	ANN	Euclidian
a	100%	100%	90%
e	100%	100%	90%
i	100%	100%	90%
o	100%	100%	80%
u	100%	100%	80%
Total	**100%**	**100%**	**86%**

Table 2. Number recognition.

Numbers	SVM	ANN	Euclidian
0	100%	100%	90%
1	100%	100%	90%
2	90%	90%	90%
3	70%	70%	60%
4	90%	90%	80%
5	60%	50%	50%
6	100%	100%	90%
7	100%	100%	80%
8	100%	100%	90%
9	100%	100%	90%
Total	**90.1%**	**90%**	**81%**

5. Conclusions

In this work we have shown that for isolated character recognition support vector machines are an efficient way of classification.

We have combined the three first Flusser moments, the compactness factor, eccentricity, size and elongation of the object as features to describe the objects. In this case we have observed that the percentage of recognition is considerable high for the five vowels and the numbers.

We have noticed that the percentage of recognition is not sufficient, because the goal is to increase it until one hundred percent, reason why still lack to find a better set of features vector that get this purpose.

References

[SVM] Support vector machines for pattern classification / Shigeo Abe, Page 83.

[SRM] V. Vapnik. Statistical Learning Theory. John Wiley & Sons, New York, 1998.

[IMF] Pattern recognition, 1993, vol. 26, pag. 167 – 174.

[LP] Handbook of character recognition and document image analysis. Pag. 35, 41-42.

[DIPE] Digital Image Processing. Rafael C. Gonzales, Pag. 416.

[DIPD] Digital Image Processing. Rafael C. Gonzales. Pag 518.

[ER] Recognition of On-line Handwritten Mathematical Formulas in the E-Chalk System. Ernesto Tapia and Raúl Rojas, Free University of Berlin Institute of computer Science Takurstr 9, D-14195 (ICDAR 2003).

[LR] Support Vector Machines for Handwritten Numerical String Recognition. Luiz S. Oliveira and Robert Sabourin, Pontificia Universidade Católica de Paraná, Curitiba, Brazil, Ecole de Technologie Supérieure – Montreal, Canada

[Stprtool] Center for Machine Perception, Department of Cybernetics Faculty of Electrical Engineering, Czech Technical University Technick´a 2, 166 27 Prague 6, Czech Republic

[OAA] Support vector machines for pattern classification, Shigeo Abe, page. 84.

[SV] Support vector machines for pattern classification, Shigeo Abe, page. 18.

[SMO] Support vector machines for pattern classification, Shigeo Abe, page. 166

Heuristics to Reduce the Training Time of SVM Algorithm

Ariel García-Gamboa, Neil Hernández-Gress, Miguel González-Mendoza, Rodolfo Ibarra-Orozco, and Jaime Mora-Vargas
ITESM-CEM, Carretera Lago de Guadalupe Km 3.5, Atizapán de Zaragoza, Estado de México, C.P. 52926, México
{ariel.garcia, ngress, mgonza, rodolfo.ibarra, jmorag}@itesm.mx
(Paper received on September 06, 2006, accepted on September 27, 2006)

Abstract. This paper presents two initialization algorithms together with the study of the Quadratic optimization problem (QP), in order to reduce the training time of Support Vector Machines (SVMs). The QP problem is very resource consuming, because the quadratic form is dense and the memory requirements grow square the number of data points. The SVM-QP problem can be solved by several optimization strategies, but for large-scale applications, a general extension consisting in the decomposition of the QP problem into smaller ones is used. The support vectors found in the training of SVMs represent a small subgroup of the training patterns. The algorithms used in this approach are to initialize the SVMs making a fast approximation of the points standing for support vectors, and finally, making the training phase only with those data. The combination of these initialization algorithms and the decomposition approach, coupled with different QP solvers specially arranged for the SVM-QP problem are compared using some well-known benchmarks.

1. Introduction

Support Vector Machines (SVMs) is a well-known technique for training separating functions in pattern recognition tasks and for function estimation in regression problems. In several problems has shown its generalization capabilities. The mathematical problem and solutions were settled by Vapnik and Chervonenkis in [1]. Training a Support Vector Machine (SVM) consists in solving a Quadratic Programming (QP) problem. While solving this QP problem, SVMs solves two problems that classical Neural Networks have: 1) choose an optimal topology and 2) adjust the parameters of the network. Since the number of variables in the QP-problem is equal to the number of training patterns, the optimization problem becomes challenging, because the quadratic form is dense and the memory requirements grow square the number of data points. Both algorithms used in this paper, used to train a SVM, are based on the idea that the support vectors represents a small subgroup of the training patterns and if we train a SVM only with those data, the same results are obtained as trained using the entire data base.

2. Support Vector Machines

Support Vector Machines is a well-known technique for solving classification, regression and density estimation [1] problems. This learning technique provides a

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 237-246

convergence to a globally optimal solution and, for several problems it has shown better generalization capabilities than other learning techniques.

In agreement with the inductive principle of the structural risk minimization, [1], in the statistical learning theory, a function, which describes correctly a training set X and which belongs to a set of functions with a low VC (Vapnik-Chervonenkis) dimension, will have a good generalization capacity independently of the input space dimension. Based on this principle, the Support Vector Machines [1] have a systematic approach to find a linear function, belonging to a set of functions with a low VC dimension.

The principal characteristic of SVMs for pattern recognition is to build an optimal separating hyperplane between two classes. If this is not possible, the second great property of this method resides in the projection of X space into a Hilbert space F of highest dimension, through a function $\varphi(x)$. One example is the internal product evaluated using kernel functions:

$$k(\mathbf{x}_i,\mathbf{x}_j) = \phi(\mathbf{x}_i)^{\mathrm{T}}\phi(\mathbf{x}_j),\ i,j = 1,...,l \tag{1}$$

satisfying Mercer conditions, like the Gaussian kernel:

$$k(\mathbf{x}_i,\mathbf{x}_j,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x_i - x_j)^2}{2\sigma^2}\right) \tag{2}$$

Thus, thanks to the freedom of using different types of kernels, the optimal separating hyperplane corresponds to different non-linear estimators in the original space.

For classification tasks, the main idea can be stated as follows: given a training data set $(\mathbf{X})$ characterized by patterns $x_i \in \Re^n, i = 1,...,n$ belonging two possible classes $y_i \in \{1,-1\}$, there exist a solution represented by the following optimization problem:

Maximize
α
$$L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) \tag{3}$$

Under the constraints
$$\sum_{i=1}^{l} y_i\alpha_i = 0, \qquad 0 < \alpha < c \tag{4}$$

where α_i are the Lagrange multipliers introduced to transform the original problem formulation with linear inequality constraints into the above representation, [1]. The parameter C controls the misclassification level on the training data and therefore the margin. The $k(x_i,x_j)$ term represents the so called kernel trick and is used to project data into a Hilbert space F of higher dimension using simple functions for the computation of dot products of the input patterns: $k(x_i,x_j) = \phi(x_i)^T\phi(x_j)$, $i,j = 1,...,n$. Once one has the solution, the decision function is defined as:

$$f(\mathbf{x}) = \mathrm{sign}\left(\sum_{i=1}^{l}\alpha_i y_i k(\mathbf{x}_i,\mathbf{x}) + b\right) \tag{5}$$

The solution to the problem formulated in (4) is a vector $\alpha_i^* \geq 0$ for which the α_i strictly greater than zero are the support vectors.

3. Quadratic programming optimization problem

Let us define the matrix Q as $Q_{ij} = y_i y_j k(x_i, x_j)$ $i, j = 1,...,l$ and the vectors $\alpha = [\alpha^1,...,\alpha^l]^T$, $1 = [1^1,...,1^l]^T$, $y = [y^1,...,y^l]^T$ and $C = [C^1,...,C^l]^T$, the SVM-QP can be written in matrix form as:

$$\underset{\alpha}{\text{Minimize}} \quad q(\alpha) = \frac{1}{2}\alpha^T Q\alpha - 1^T\alpha \tag{6}$$

$$\text{Under the constraints} \quad y^T\alpha = 0, \quad 0 \leq \alpha \leq C \tag{7}$$

The global minimum of α is guaranteed because of the problem's convexity.

3.1. Optimality conditions

The QP problem must satisfy first and second order conditions, but Ec (6-8) make any first order critical point a global solution, [8]. In order to solve the problem, the classical Lagrange duality theory is used and then we have: $\gamma \in \Re$ to (7) and $\beta = [\beta^1,...,\beta^l]^T$ and $\chi = [\chi^1,...,\chi^l]^T$ to (7) the primal lagrangian is:

$$L_p(\alpha,\beta,\chi,\gamma) = \frac{1}{2}\alpha^T Q\alpha - 1^T\alpha + \gamma y^T\alpha - \beta^T\alpha + \chi^T(\alpha - C) \tag{8}$$

where α has to be minimized. Thus, the stationary conditions are:

$$y^T\alpha = 0, \alpha \geq 0 \text{ and } C - \alpha \geq 0 \quad \text{(primal feasibility)} \tag{9}$$

$$Q\alpha - 1 + \gamma y - \beta + \chi = 0, \ \beta \geq 0 \text{ and } \chi \geq 0 \quad \text{(dual feasibility)} \tag{10}$$

$$\beta^T\alpha = 0 \text{ and } \chi^T(\alpha - C) = 0 \quad \text{(complementary conditions)} \tag{11}$$

which are linear functions of α, γ, β and χ. Thus, one can obtain the solution of the SVM-QP problem by finding a non negative solution for the l equations (9) which also satisfies the l equations (9-11). Also, the dual of the QP problem by:

$$\underset{\alpha}{\text{Minimize}} \quad L_D(\beta,\chi,\gamma)=\frac{1}{2}(1-\gamma y+\beta-\chi)^T Q^{-1}(1-\gamma y+\beta-\chi)-\chi^T C \tag{12}$$

$$\text{Under the constraints} \quad \beta \geq 0, \tag{13}$$

$$\chi \geq 0, \tag{14}$$

which has the same KKT conditions and solution.

4. Algorithms for quadratic optimization problems

There are two main types of algorithms for the resolution of QP problems:

- Interior point methods that aim for the complementary conditions by keeping primal and dual feasibility at the same time.
- Active set methods, which are divided in primal and dual. Primal methods aim for dual feasibility by keeping primal feasibility and complementary conditions and the dual methods work with primal feasibility by keeping dual feasibility and the complementary conditions. The dual methods have the condition that the Q matrix must be a positive definite matrix.

5. Large scale SVM-QP implementation

Optimization algorithms to solve the SVM-QP (6)-(7), described in the preceding section are operational to solve problems of less than 2000 examples. Beyond this limit, depending on memory and processing capacities, it is not possible to use any QP technique without some modifications. For large-scale data bases, it is difficult to calculate and store the matrix Q of the vector products $k(x_i,x_j)$, because of data processing limitations. Consequently, one must find more effective methods for this kind of problems and be able to find the optimal solution in the minimum time with a moderate request of data-processing resources.

5.1. Decomposition technique

Different authors, [1], [9], [10] and [11], introduced the idea to break up the SVM-QP problem, into smaller sub problems which are easier to deal with. All those strategies consider two key points:

- Optimality conditions that make possible to check if the algorithm has optimally solved the problem, for the SVM-QP problem and its optimality conditions.
- Strategy of implementation, that defines the implementation of the objective function associated with variables violating optimality conditions, if a particular solution is not the global solution.

The solution of the SVM-QP (6)-(7) is optimal if and only if KKT conditions (9-11) are satisfied knowing that matrix Q is semi-definite positive. KKT conditions have a simple form to check them, thus, the SVM-QP problem can be solved when for any $i = 1,...,l$:

$$\alpha_i = 0 \rightarrow y_i g(\mathbf{x}_i) > 1 \quad (15)$$

$$0 < \alpha_i < C \rightarrow y_i g(\mathbf{x}_i) = 1 \quad (16)$$

$$\alpha_i = C \rightarrow y_i g(\mathbf{x}_i) < 1 \quad (17)$$

with $g(x_i)$ the argument of the sign decision function (5):

$$g(\mathbf{x}_i) = \sum_{j=1}^{l} \alpha_j y_j k(\mathbf{x}_j, \mathbf{x}_i) + b \quad (18)$$

In order to incorporate optimality conditions, implementation strategy must take into account the fact that a significant part of Lagrange multipliers of α_i are equal to zero in the solution. In a similar way to primal active set methods, a solution is to divide the training set in active set A, also known as working set, and its complement N. Then, it is possible to rewrite the SVM-QP problem as follows:

$$\underset{\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_N}{\text{Minimize}} \quad q(\boldsymbol{\alpha}_A, \boldsymbol{\alpha}_N) = \frac{1}{2}\begin{bmatrix}\boldsymbol{\alpha}_A \\ \boldsymbol{\alpha}_N\end{bmatrix}^T \begin{bmatrix}\mathbf{Q}_{AA} & \mathbf{Q}_{AN} \\ \mathbf{Q}_{NA} & \mathbf{Q}_{NN}\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_A \\ \boldsymbol{\alpha}_N\end{bmatrix} - \begin{bmatrix}\mathbf{1}_A \\ \mathbf{1}_N\end{bmatrix}^T \begin{bmatrix}\boldsymbol{\alpha}_A \\ \boldsymbol{\alpha}_N\end{bmatrix}$$

$$\text{Under the constraints} \quad \begin{bmatrix}\mathbf{y}_A \\ \mathbf{y}_N\end{bmatrix}^T \begin{bmatrix}\boldsymbol{\alpha}_A \\ \boldsymbol{\alpha}_N\end{bmatrix} = 0,$$

$$\begin{bmatrix}\mathbf{0}_A \\ \mathbf{0}_N\end{bmatrix} \le \begin{bmatrix}\boldsymbol{\alpha}_A \\ \boldsymbol{\alpha}_N\end{bmatrix} \le \begin{bmatrix}\mathbf{C}_A \\ \mathbf{C}_N\end{bmatrix},$$

in which we can replace any $i \in A$ by any $j \in N$, without modifying the cost function. The main idea is to have only the support vectors in the active set A.
In this manner, the inactive set N is formed by zero-multipliers α_N.

Algorithm 1. Decomposition algorithm of the SVM-QP.

1. Election of an active unit initial A of size n_A.
2. Solve the QP (6)-(7), defined by active set A.
3. While there exist any $j \in N$ violating $y_j g(\mathbf{x}_j) > 1$.
   a. shift the n_A most erroneous vectors $\mathbf{x}_j$ to active set A,
   b. shift all vectors $\mathbf{x}_i$ with $\alpha_i = 0$, $i \in A$, to inactive set N, and return to step 2

The solution of this problem will be the solution of the SVM-QP problem if it verifies the optimality conditions (15)-(17), in particular $j_j g(x-j) > 1$, $j \in N(\alpha_j = 0)$. If it is not the case, then α_j, which corresponds to x_j, must be different to zero and, consequently, it is necessary to shift it to active set A. Certainly, we can make the same

for vectors x_i, associated with $\alpha_i = 0$ to N. The algorithm (1) summarizes the decomposition algorithm of the SVM-QP problem. To introduce better than a random initial working set, we can use a Gaussian search and accelerate the learning process [12]. An extreme of the decomposition is the sequential minimal optimization, SMO, suggested by Plat, [11], but, it is not useful for our work since it finds a sub optimal solution.

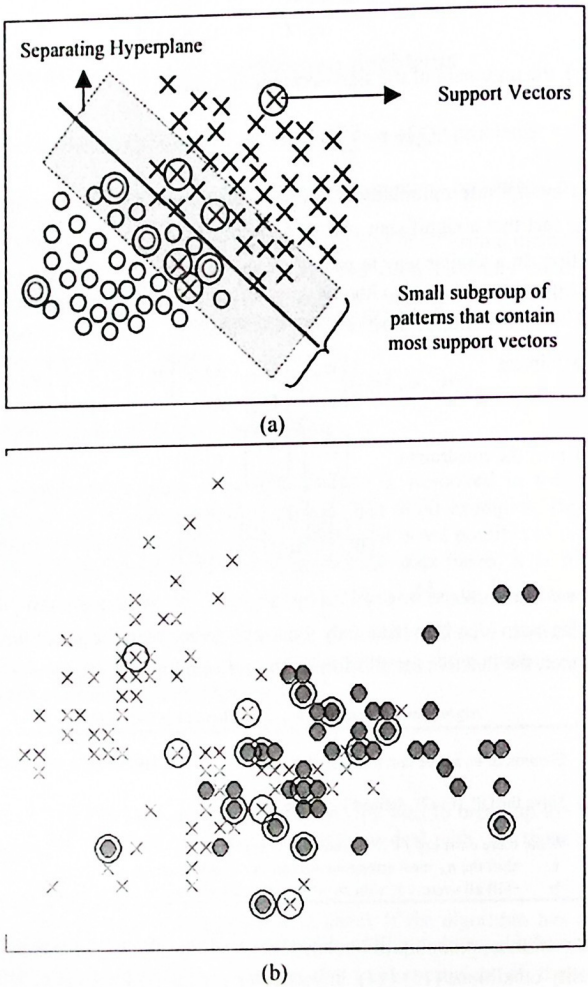


Fig. 1. Initialization strategies. (a) BCP initialization (b) Kernel Perceptron initialization.

6. Initialization strategies

Solving large scale applications require a decomposition method that breaks up the original problem into smaller QP sub problems. The main disadvantage with this approach is that patterns for each sub problem are selected randomly (causing substantial difference in the learning rate) for the decomposition method. In order to obtain a better initialization working set (in order to accelerate the learning process) algorithms like Kernel-Perceptron and BCP are used, instead of using a random one. To perform the task we take advantage of the characteristics of these algorithms: find an optimal separating hyperplane for a given dataset, easy to implement and minimal processing and memory requirements. The combination between these algorithms and the decomposition approach ensures that the initial sub problem is conformed by candidate support vectors that were extracted before training the SVM, and consequentially, the training time is reduced. The general idea is to obtain a hyperplane well-classifying the original dataset, by means of the proposed algorithms, then, by simple Euclidean distance between the hyperplane w and the different patterns x, we get the closest to w and form the active set A. Due to geometrical characteristics of the clusters, we can demonstrate that if the length of A is greater enough, then support vectors are included, and thus, we have the best working set A to train the SVM (Refer to Figure 1).

6.1. Barycentric Correction Procedure

BCP is an algorithm based on geometrical characteristics for training a threshold unit [2]. It is very efficient training linearly separable problems and it was proven that the algorithm rapidly converges towards a solution [3]. Also, it is a free parameters algorithm offering very fast solutions for any kind of problems. The algorithm defines a hyperplane $\mathbf{w}^T \cdot \mathbf{x} + \theta = 0$ dividing the input space for each class. Thus, we can define: $I_1 = \{1,\ldots,N_1\}$ and $I_0 = \{1,\ldots,N_0\}$ where N_1 represents the number of patterns of target 1 and N_0 the number of patterns of target -1. Also, let $b = (b_0, b_1)$ be the barycenter of points C_1 and C_0, weighted by the positive coefficients $\alpha = (\alpha_1,\ldots,\alpha_{N_1})$ and $\mu = (\mu_1,\ldots,\mu_{N_0})$ referred as weighting coefficients [3]:

$$b_1 = \frac{\sum_{i \in I_1} \alpha_i p_i}{\sum_{i \in I_1} \alpha_i}, \quad b_0 = \frac{\sum_{i \in I_0} \mu_i m_i}{\sum_{i \in I_0} \mu_i} \tag{19}$$

The weight vector w is defined as a vector difference $\mathbf{w} = b_1 - b_0$. At each iteration, barycenter moves towards misclassified patterns. Increasing the value of particular barycenter implies hyperplane moves on that direction. For computing the bias term θ, let's define $\vartheta : \Re^n \to \Re$ such that $\vartheta(p) = -\mathbf{w} \cdot p$. The bias term is calculated as follows: $\theta = \frac{\max \vartheta_1 + \min \vartheta_0}{2}$ Assuming the existence of $J_1 \subset I_1$ and $J_0 \subset I_0$

that refer to misclassified examples of target (+1, -1), barycenter modifications are calculated by:

$$\forall_i \in J_1 \qquad \alpha(new)_i = \alpha(old)_i + \beta_i \tag{20}$$
$$\forall_i \in J_0 \qquad \mu(new)_i = \mu(old)_i + \delta_i$$

Where β and δ are defined as follows:

$$\beta = \max\left\{\beta_{--}, \min\left[\beta_{-}, \frac{N_1}{N_0}\right]\right\}, \quad \delta = \max\left\{\delta_{--}, \min\left[\delta_{-}, \frac{N_0}{N_1}\right]\right\}$$

According to [3], β_{--} and δ_{--} can be set to 1 and β_{-}, δ_{-} set to 30.

6.2. Perceptron algorithm

The Perceptron algorithm [5] is a first approach method to deal with linearly separable problems. It is an incremental algorithm that starts with a weight vector $\mathbf{w} = \mathbf{0}$. At each iteration, small modifications to $\mathbf{w}$ are performed until a solution is reached. Convergence is ensured in a finite number of iterations for linearly separable problems. In this research, we make some modifications to the original algorithm to treat non-linearly separable problems.

Algorithm 2. Perceptron algorithm.

1. Initialize the weight vector $\mathbf{w}$, b and choose a learning step η
2. While there exist $i: i \in N$ such that $f(\mathbf{x}_i) \neq y_i$
3. Update w and b values according to $\Delta w = w_{old} + (\eta/2)(y_i \cdot \mathbf{x}_i)$ and $\Delta b = b_{old} + (\eta/2)(y_i)$

6.3. Kernel Perceptron extension

The kernel Perceptron algorithm [6] deals with non-linearly separable datasets. Basically the algorithm defines the dual function: $f(x) = \sum_{i=1}^{N} \gamma_i y_i \{\phi(\mathbf{x}^T_i) \cdot \phi(\mathbf{x})\} + b$ Where γ is the set of dual variables to be updated and the dot product $(\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}))$ is replaced for the kernel function $k(\mathbf{x}_i, \mathbf{x})$.

7. Experimental Results

In this section, we present the comparison tables of the two initialization strategies and the two adapted QP-SVM implementations: a dual active set algorithm (*QP1*) and an interior point algorithm (*QP2*). Also, a comparison of these methods with a second interior point method called pr-loqo (*QP3*), [4] is presented. For every initialization

strategy, we coupled a decomposition algorithm: *BCP +QP*, and *KernelPerceptron+QP*,. We used seven benchmark problems commonly referred in the literature: iris (iri), sonar (son), Pima Indian diabetes (pid), Tic Tac Toe (tic), Phonemes (pho), adult (adu) and Shuttle (shu) data sets [7]. Experiments were done with a RBF function with $\gamma = 0.5$ for the kernel parameters and the regularization parameter C was set to 1000. These parameters were selected by means of cross-validation. Results shown in Tables 1 and 2 are the standard mean of about 1000 executions. The best results were obtained using the *QP1* approach and the two initialization strategies (BCP and Kernel Perceptron). Although the good results obtained, an important problem is noted: The *BCP + QP* strategy has a good performance with these data sets but, if a benchmark with a high degree of non-linearity is tested, it is not guaranteed that the initial working set contains a significant number of support vectors. It is because the approach is approximating the solution by a linear function (in the original space), while the solution of the SVM is reached in a high dimensional space (in feature space). The *KernPerceptron+QP* strategy does not have this problem, because the Kernel Perceptron algorithm looks for the solution in the same high dimensional space that the SVM algorithm.

Table 1. Time performance of BCP initialization and QP solvers

	N	BCP + QP1	BCP + QP2	BCP + QP3
iri	150	0.0123 sec	0.391 sec	5.201 sec
son	208	0.16 sec	6.213 sec	16.231 sec
pid	768	5.143 sec	25.450 sec	42.960 sec
tic	958	3.091 sec	28.456 sec	41.975 sec
pho	1027	3.328 sec	26.727 sec	43.150 sec
adu	5000	24.028 sec	129.2773 sec	348.152 sec
shu	13633	35.431 sec	92.692 sec	202.428 sec

Table 2. Time performance of Kernel Perceptron initialization and QP Solvers

	N	Kperceptr + QP1	Kperceptr + QP2	Kperceptr + QP3
iri	150	0.0136 sec	0.3673 sec	6.010 sec
son	208	0.1231 sec	5.878 sec	14.231 sec
pid	768	2.296 sec	22.086 sec	38.069 sec
tic	958	1.635 sec	24.876 sec	40.352 sec
pho	1027	3.250 sec	28.297 sec	53.150 sec
adu	5000	13.35 sec	128.301 sec	355.278 sec
shu	13633	19.825 sec	102.192 sec	262.628 sec

8. Conclusions

Support Vector Machines is a promising methodology used in different research areas. Moreover, the optimization of the SVM is a delicate problem due to computational and memory requirements. This research is focused in combining different initialization strategies and three different SVM-QP solvers in order to select one that improves the training time of SVMs. The comparison of the different proposals shows that using kernel Perceptron and BCP as initialization strategies has a good performance, but the

SVM-QP solver that minimizes the training time was *QP1* which corresponds to the dual active set algorithm. Although, it was shown that the *BCP* initialization could have limitations when training data sets with a high non-linearity degree.

References

[1] Vapnik, V. Computational Learning Theory. John Wiley & Sons (1998)
[2] Poulard, H., Stève, D. Barycentric Correction Procedure: A fast method of learning threshold unit. World Congress on Neuronal Networks I (1995) 710-713
[3] Poulard, H., Stève, D. A convergence theorem for Barycentric Correction Procedure. Technical Report 95180 of LAAS-CNRS (1995)
[4] González-Mendoza, M. Quadratic Optimization fine tuning for the learning phase of SVM. ISSADS 2005 11 (2005).
[5] Rossemblat, F. The Perceptron: a probabilistic model for information storage and organization in the brain. Psychological review (1958) 386-408.
[6] Kowalczyk, A.: Maximal margin Perceptron. MT Press, Cambridge (1999).
[7] Keogh, E., Blake, C. and Merz, C.J. UCI repository of machine learning databases. http://kdd.ics.uci.edu (1998).
[8] Fletcher, R. Practical Methods of Optimization John Wiley and Sons; 2nd edition, May (2000).
[9] Joachims, T. Making large-scale support vector machine training practical. In Advances in Kernel Methods: Support Vector Machines. B. Schlkopf, C. Burges, A. Smola editors, MIT press, Cambridge, MA, pp. 169-184. (1998).
[10] Osuna, Edgar. Support Vector Machines: Training and Applications. Report of the Center of Biological and Computational Learning MIT. Paper No. 144 (1997).
[11] Platt, J. C. Fast Training of support vector machines using sequential minimal optimization. In B. Schlkopf, C. J. C. Burges and A. J. Smola, editors. Advances in Kernel Methods Chapter 12, pages 185-208. MIT press (1998).
[12] González-Mendoza, M, Titli, A., Hernández-Gress, N. Boosting Support Vector Machines in Density Estimation Problems. IEEE Information Processing and Management of Uncertainty, IPMU 2002. Annecy, France, (2002).

Vision and Image Processing

Performance of Lossy Image Compression Coders on Textures

Osslan Osiris Vergara Villegas[1], Raúl Pinto Elías[1] and José Ruiz Ascencio[1]
[1] Centro Nacional de Investigación y Desarrollo Tecnológico (cenidet)
Computer Science Department
Interior internado Palmira s/n Col. Palmira Cuernavaca Morelos México
{osslan, rpinto, josera}@cenidet.edu.mx
(Paper received on August 11, 2006, accepted on September 26, 2006)

Abstract. Image compression algorithms should be evaluated by at least two important criteria: the compression rate and the quality of the reconstructed images. Other factors which may be considered are compression/decompression times, the memory requirements, etc. This paper shows the results obtained in a performance comparison between six lossy compression algorithms on texture images. We measure the quality of the reconstructed image and the compression/decompression process time at moderate and low bit rate compression ratios. The results shown allow the selection of an algorithm as a function of image texture, in future we intend to measure other image components such as edges and edge-associated detail to have the possibility of recommending a compression algorithm that adapts to any image that we need to compress.

1. Introduction

Digital images are much used in several disciplines of computer science besides Digital Image Processing (DIP). A great quantity of the information that human beings use daily is in a digital form, digital images need a great amount of storage space and more bandwidth and time to transmit it for example via Internet. A possible solution to handle this data and to use less storage space and time to transmit is to use image compression.

The goal of image compression is to reduce the bit rate for signal transmission or storage while maintaining an acceptable image quality for various purposes [1]. The compression process can be made with or without loss of information. In lossless methods the decompressed data are an exact copy of the originals, whereas in the lossy methods, the decompressed data are an approximation of the originals.

Image compression methods have been evaluated on the basis of minimizing an objective distortion measure at a given level of data compression [2]. At low bit rates an image can be distorted or present artifacts in components such as edges or textures and it becomes imperative to measure the performance of a particular coder in order to know if there exists a balance between the quality of the reconstructed image and the compression factor obtained, which is a suitable criterion in lossy compression coders.

In the literature there are a few works [3], [4], [5] that attempt to measure the performance of a compression algorithms in order to try to recommend an adequate algo-

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 249-258

rithm for the image that needs to be compressed. The difference between this paper and others reported in the literature is that we present the first and novel results obtained in a comparison made using the very important information of a texture cue in an image. We made the comparison by measuring the time used to compress/ decompress an image and the quality of the reconstructed image at moderate and low bit rate. This study was made as a base work in order to design in the future a compression coder with feature preserving (textures, edges, and edge associated detail).

In the following sections we present a brief description of the proposed model and the tests and results obtained from the compression algorithms comparison.

2. Proposed Model

All the objects in the real world have a surface that can reflect light in a way that depends on the structure of the surface. That manner of reflecting light is known as the visual texture of the objects and gives information about the material that the object is composed of (wood, water, steel, wool, etc) and some properties (roughness, regularity, brightness, homogeneity, etc) that inform about the state (wet, clean, liquid, etc) of the object.

An image gives us the sensation of texture by the repetition of similar patterns, in [6] Julez gives a definition of texture: Textures are composed by a small number of similar types of atoms called textons, that are repeated in almost regular or random position and orientation.

Several authors agree that the three most important dimensions for texture perception by human beings are: periodicity, randomness and directionality [7], [8]. A texture image can be classified in one of the three mentioned types and one can use the Fourier spectrum to help the task of classification [7].

A rough description of the Fourier spectrum for each texture class is:

a) *Periodic textures:* The spectrum consists of significant peaks scattering out regularly in some directions.
b) *Directional textures:* In the Fourier spectrum the directionality will be preserved.
c) *Random textures:* The distribution of the responses of the spectrum is not restricted to certain directions.

In this paper we measure the performance of lossy image compression algorithms on each of the three classes of textures with the goal of investigating which compressor has the best performance for each class in order to offer the possibility of recommending a compression algorithm for the particular texture information of an image.

The proposed model has six phases, first we select the database of image textures, second we compute the Fourier spectrum, third we classify the image in one of the three texture classes, fourth the image is compressed/decompressed with six lossy image coders, fifth the error between original and reconstructed image and the time used to compress/decompress are measured; and finally we make comparisons between the images. Figure 1 shows the phases for the proposed model.

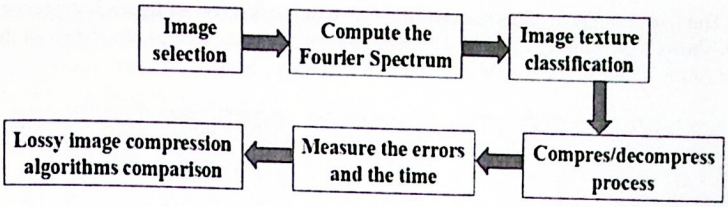


Fig. 1. The six modules of the proposed model.

2.1. Image Selection, Fourier Spectrum Computing and Texture Classification

We obtain the texture image database from the Brodatz album [9]. The album has become a standard for evaluating texture algorithms and consists of 112 texture images belonging to different classes. We select five images for each texture class for a total of 15 texture images that were used for the tests in this paper. The selection of the textures for each class was made by computing the Fast Fourier Transform (FFT) equation 1 and the Fourier Spectrum (FS), equation 2.

$$F(p,q) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m.n) e^{-J(2\pi/M)} e^{-J(2\pi/N)qm} \tag{1}$$

$$|F(u,v)| = \left[R^2(u,v) + I^2(u,v)\right]^{1/2} \tag{2}$$

By means of visual analysis of the FS we are ensured that the images selected belong to the texture class attributed to them. The selected images are greyscale images (256 x 256) which need a storage space of 66Kb (65536 bits). After the analysis five directional textures were selected showed in the first row of figure 2, the second row shows their correspondent Fourier Spectra.

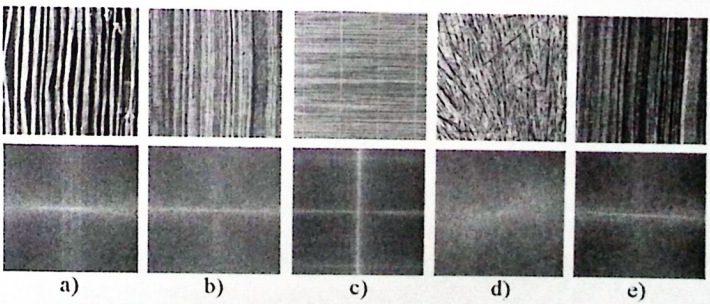


Fig. 2. Directional textures. a) D51 raffia woven with cotton threads, b) D106 cheesecloth, c) D49 Straw screening, d) D15 straw and e) D105 cheesecloth.

The five periodic textures selected are shown in the first row of figure 3, the second row shows their Fourier Spectra. The five random textures selected are shown in the first row of figure 4, the second row shows their Fourier Spectra.

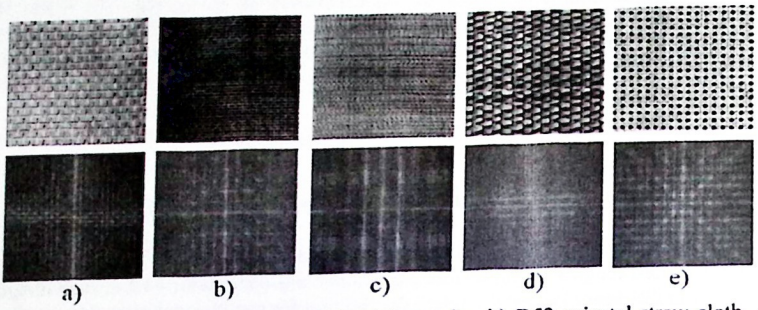


Fig. 3. Periodic textures. a) D1 woven aluminium wire, b) D52 oriental straw cloth, c) D55 straw matting, d) D56 straw matting and e) D101 cane.

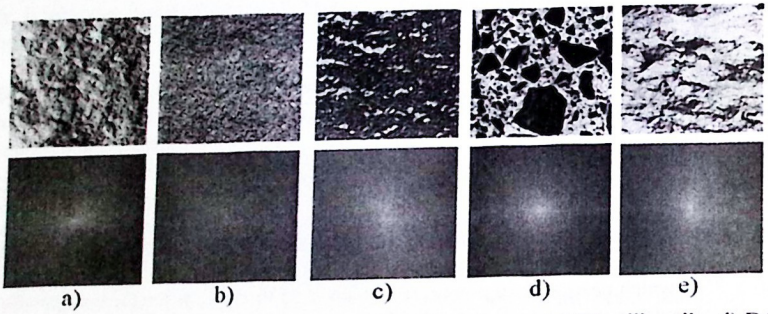


Fig. 4. Random textures. a) D2 fieldstone, b) D9 grass lawn, c) D86 ceiling tile, d) D62 european marble and e) D7 fieldstone D2.

2.2. Compress/Decompress Process

Once we have the textures classified adequately, the compress/decompress phase of the images is carried out. We select six lossy image compression algorithms: Discrete Cosine Transform (DCT) [10], Haar Wavelet Transform (HWT) [11], Singular Value Decomposition Transform (SVDT) [10], Daubechies 4 Wavelet Tranform (D4WT) [11], Embedded Zerotree Wavelet (EZW) [12] and Set Partitioning In Hierarchical Trees (SPIHT) [13]. These algorithms were programmed as proposed in the cited references and were set to obtain similar bit rates so that all coders have the same conditions and we avoid skewing the results.

Compression standards such as JPEG and JPEG2000 were not used because this work intends to serve as the basis of a future image coder with feature preservation

with respect to a target pattern recognition algorithm. A disadvantage of standards is that the only reference is human vision and they attempt to equally preserve the quality of any image. Thus they can not be easily manipulated for tests cases, and at very low bit rates the images obtained are corrupted by artifacts, and image features such as edges may be practically destroyed.

The first compress/decompress process was made with a bit rate of 0.5 with the resulting images using 33Kb (32768 bits). After the process we have a total of 90 texture images (30 in each class). The second process was made using a bit rate of 0.1, with resulting images using 6Kb (6554 bits). Its purpose was to find the algorithm with the best performance in our worst case scenario. Here we again obtain 90 texture images.

2.3. Measuring the Errors and Time

After the lossy compress/decompress process, it is important to measure the difference between the original image (I) and the decompressed image (I') in order to determine the quality of the reconstructed image [14]. For this purpose we use four well known objective measures: Mean Square Error (*MSE* eq. 3), Peak Signal to Noise Ratio (*PSNR* eq. 4) and Norm 2 and Frobenius relative errors (eq 5).

$$MSE = \frac{1}{MN}\sum_{y=1}^{M}\sum_{x=1}^{N}\left[I(x,y)-I'(x,y)\right]^2 \tag{3}$$

$$PSNR = 10 * \log_{10}(255 / sqrt(MSE)) \tag{4}$$

$$N_2, F = \frac{\left\| I - I' \right\|_{2,F}}{\left\| I \right\|_{2,F}} \tag{5}$$

For the measurement of time we implement a function in every algorithm to add up the time in seconds needed to complete the process of compressing/decompressing an image of a texture. To avoid skewing the results, all codecs were programmed by the same programmer, in C++ Builder, and executed in a Windows XP environment running on Intel Pentium IV processor at 2.3 GHz. with 256 MB of RAM memory.

2.4. Lossy Image Compression Algorithms Comparisons

In this stage it is important to analyze which texture of each class represents more problems for the compression algorithms; that is, which texture is better reconstructed, and which texture has poor reconstruction for each algorithm.

On the other hand, it is important to analyze the performance of the coders regarding execution time, it is desirable that the process have a balance between the quality of the reconstructed image and the time used to compress/decompress an image.

With these two measures we can compare the performance of the algorithms for each texture class and then give a recommendation of the best algorithm.

3. Tests and Results

We have two test groups, reconstructed images obtained from: a) 0.5 bit rate and b) 0.1 bit rate as pointed out in section 2.2. We have 15 texture images on which we apply six compression algorithms to obtain a total of 90 images for case *a* and 90 images for case *b*, for a total of 180 texture images.

For each image we compute the four error measures proposed in section 2.3 and the time used for the compress/decompress process of an image. As a result, we have five measures for each of the 180 images. Because of the limited space of this paper we only show the mean and the standard deviation results obtained for each texture class for each bit rate. Figure 5 shows an example of different coders reconstructed images: the first row show 0.5 bit rate images and second row 0.1 bit rate images.

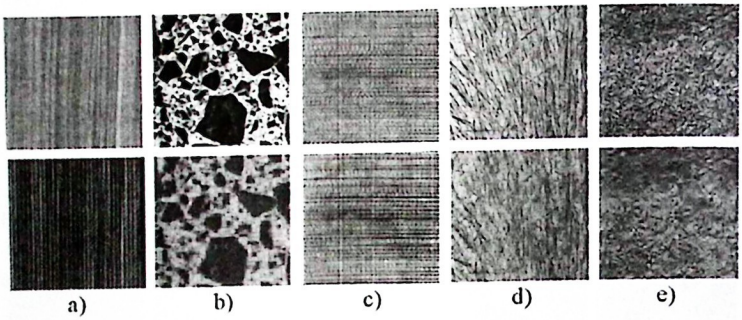


Fig. 5. Reconstructed texture images. a) HWT D105, b) EZW D62, c) DCT D55, d) D4WT D15, e) SPIHT D9.

Table 1 shows the results for the mean (upper half) and for the standard deviation (lower half) for directional textures, table 2 the results for periodic textures and table 3 the results for random textures all with bit rate of 0.5.

Table 1. Directional textures average and standard deviation results for bit rate 0.5.

Average					
Compression algorithm	*MSE*	*PSNR*	*Frobenius*	*Norm2*	*Time in seconds*
DCT	191.6071	26.6042	0.0919	0.0568	1.51
HWT	1616.7638	16.4300	0.2976	0.2096	2.102
SVDT	147.5686	28.0624	0.0789	0.0364	6.0028
D4WT	427.6316	23.0812	0.1395	0.0713	3.31
EZW	83.0009	29.5485	0.0642	0.0321	11.8
SPIHT	630.0550	21.2940	0.1693	0.0474	7.92
Standard deviation					
DCT	154.6206	3.8576	0.0279	0.0402	0.0761
HWT	595.6014	2.3081	0.0896	0.1115	0.0634
SVDT	172.6100	3.6981	0.0319	0.0127	0.2633
D4WT	303.4030	4.0181	0.0444	0.0470	0.0894
EZW	57.7073	2.3492	0.0138	0.0061	0.2121
SPIHT	505.4763	3.5089	0.0509	0.0357	0.1303

Table 2. Periodic textures average and standard deviation results for bit rate 0.5.

Compression algorithm	*MSE*	*PSNR*	*Frobenius*	*Norm2*	*Time in seconds*
			Average		
DCT	248.5560	25.6067	0.1230	0.0837	1.46
HWT	2341.2887	15.3872	0.3884	0.3312	2.186
SVDT	101.3690	28.5201	0.0822	0.0647	5.978
D4WT	478.2486	22.0153	0.1682	0.1150	3.252
EZW	87.9482	29.020	0.1282	0.0940	11.722
SPIHT	659.4961	20.5227	0.1950	0.0745	7.98
			Standard deviation		
DCT	178.8439	4.4860	0.0843	0.0798	0.1534
HWT	1674.9572	3.2333	0.2936	0.3317	0.2745
SVDT	46.3382	2.3249	0.0505	0.0453	0.2278
D4WT	315.3719	2.6695	0.0850	0.0822	0.3582
EZW	2433.3408	4.4780	0.1129	0.1154	0.4748
SPIHT	399.1214	2.4817	0.0807	0.0388	0.7049

Table 3. Random textures average and standard deviation results for bit rate 0.5.

Compression algorithm	*MSE*	*PSNR*	*Frobenius*	*Norm2*	*Time in seconds*
			Average		
DCT	221.7494	25.3617	0.1059	0.0466	1.54
HWT	1490.8119	16.6299	0.2816	0.1609	2.326
SVDT	295.1127	23.8196	0.1241	0.0378	6.24
D4WT	473.9257	21.5967	0.1579	0.0655	3.474
EZW	152.4267	27.0863	0.0879	0.0319	12.004
SPIHT	558.3568	21.0256	0.1697	0.0259	8.32
			Standard deviation		
DCT	138.6217	2.7676	0.0461	0.0228	0.0674
HWT	543.3977	1.5999	0.1047	0.1304	0.2911
SVDT	139.6775	2.0516	0.0475	0.0276	0.2302
D4WT	160.3059	1.6038	0.0481	0.0304	0.4407
EZW	81.4840	3.2836	0.0397	0.0154	0.5020
SPIHT	270.5800	1.9376	0.0586	0.0080	0.4919

For the results shown in tables 1, 2 and 3 we can give the following comments: For the three texture classes, and taking into account the quality of the reconstructed image (error measures), the EZW algorithm has the best performance. This can be seen, for example, in that MSE always has the smallest value and the PSNR is always the largest, which indicates that the images are quite close to the originals.

The next best compressor is DCT, which is demonstrated by the Frobenius and Norm2 measures, which are just above those for EZW. After DCT, we obtained the best performance with SPIHT and SVDT algorithms, in that order. The poorest performance is presented in HWT: for example in the MSE we observe that the image is reconstructed with a large difference for the three texture classes. The second worst algorithm is D4WT, it is only superior for reconstruction to the HWT algorithm.

For the time variable, different things can be said. For example the EZW algorithm which has the best performance in image reconstruction has the disadvantage of using at least twice the time to compress/decompress compared with the other algorithms. The second slowest algorithm is SVDT followed by the SPIHT algorithm.

The fastest algorithms are DCT, HWT, D4WT in that order. DCT is the fastest algorithm and if we observe that it gives a good reconstruction of the image, we can understand why DCT is a very important part of the JPEG standard.

Regarding individual textures, those that pose more problems for the compression algorithms are the directional texture D15, the periodic textures D1 and D101 and the random textures D2, D9 and D86. For the evaluation by class we can say that for the case of directional textures the best algorithm is EZW followed by SVDT. The poorest performance is presented by the HWT algorithm. For periodic textures the best algorithm is EZW followed by DCT. The poorest performance is presented by the SVDT. Finally for the case of random textures the best algorithm is the DCT followed by the SVDT. The poorest performance is presented by the HWT.

The results obtained for the case of 0.1 bit rate are shown in table 4 for directional textures, table 5 for periodic textures and table 6 for random textures.

Table 4. Directional textures average and standard deviation results for bit rate 0.1.

			Average		
Compression algorithm	*MSE*	*PSNR*	*Frobenius*	*Norm2*	*Time in seconds*
DCT	2967.4880	14.8655	0.3799	0.2496	1.172
HWT	6690	10.1215	0.5935	0.5041	1.88
SVDT	2819.9901	16.3922	0.3537	0.2215	9.8
D4WT	1760	16.2352	0.2990	0.1733	3.116
EZW	2590	14.5630	0.3813	0.2500	6.616
SPIHT	1430	17.4993	0.2596	0.1031	5.232
			Standard deviation		
DCT	2100.6320	4.4649	0.1793	0.1705	0.1285
HWT	2533.5941	1.6230	0.0577	0.0802	0.1923
SVDT	2328.7143	6.5911	0.2205	0.1986	0.2318
D4WT	899.0717	2.5749	0.0655	0.0867	0.1152
EZW	1405.9375	2.6184	0.1755	0.2067	0.5896
SPIHT	1018.8797	3.2087	0.0659	0.0802	0.2783

Table 5. Periodic textures average and standard deviation results for bit rate 0.1.

			Average		
Compression algorithm	*MSE*	*PSNR*	*Frobenius*	*Norm2*	*Time in seconds*
DCT	4950	11.3217	0.4815	0.3753	1.1808
HWT	8490	8.7890	0.6235	0.5396	1.842
SVDT	5673.289	12.68	0.4550	0.3684	9.884
D4WT	2003.4994	13.3643	0.3576	0.2779	3.072
EZW	3300	12.9024	0.4068	0.2996	7.006
SPIHT	1590	14.2571	0.2934	0.1462	5.278
			Standard deviation		
DCT	4555.6807	6.4979	0.1606	0.1382	0.2266
HWT	5536.5457	5.3933	0.0540	0.0805	0.2651
SVDT	6031.6913	8.5941	0.2298	0.1899	0.4086
D4WT	1263.4135	7.5819	0.2333	0.2585	0.3434
EZW	3556.1362	7.0534	0.2103	0.2248	0.1717
SPIHT	1103.1317	7.8059	0.0923	0.0614	0.3720

Table 6. Random textures average and standard deviation results for bit rate 0.1.

Compression algorithm	*MSE*	*PSNR*	*Frobenius*	*Norm2*	*Time in seconds*
			Average		
DCT	4345.1877	13.3438	0.4457	0.2277	1.4956
HWT	8740	9.0543	0.6464	0.5368	2.1772
SVDT	4440	12.8528	0.4538	0.2351	10.304
D4WT	1371.1854	16.8818	0.2679	0.0761	3.344
EZW	3130	14.2036	0.368	0.1826	7.1648
SPIHT	1270	17.141	0.2599	0.0509	5.613
			Standard deviation		
DCT	3221.4480	4.6856	0.2273	0.1829	0.1968
HWT	4059.253	1.8705	0.0778	0.0576	0.4515
SVDT	3109.3429	3.9110	0.1973	0.1557	0.5642
D4WT	367.7127	1.1482	0.0675	0.0319	0.2854
EZW	2818.5531	3.0621	0.1062	0.0920	0.5658
SPIHT	236.4950	0.7844	0.0661	0.0145	0.4867

For our worst test case with results in tables 4, 5 and 6 there are several comments: First the reconstructed images are obviously reconstructed with a large error. Second, the relative performance of the algorithms is different compared with the previous test case. Taking into account the quality of the reconstructed image, the poorest performance is presented by the HWT, it has the largest MSE and the reconstructed images are visually different. The next poorest performance is DCT and SVDT respectively. The best performance is given by SPITH followed by D4WT and EZW in that order.

With respect to time, the best algorithm is the DCT follow by transform algorithms HWT and D4WT. The poorest performance is presented by the SVDT algorithm. In this case we observe that the time needed to compute using less singular values is larger instead of smaller than the first test case. Next in speed are the algorithm SPIHT and EZW algorithms, in that order. Regarding individual textures, those that present problems for the compression algorithms are the directional textures D49 and D15, the periodic textures D55 and D56 and the random textures D62 and D7. For the evaluation by texture class, the best algorithm for directional textures is SVDT but it has the problem of time and the poorest is HWT. For periodic textures the best is SPIHT and the poorest is HWT. For random textures the best is DCT and the poorest is HWT.

4. Conclusions

In this paper we present the results obtained from a performance comparison of six lossy image compression algorithms. The algorithms were compared using texture images and measuring by two variables, the quality of the decompressed image and the time used for the full cycle of compression/decompression on an image. A performance ranking of the algorithms was obtained for each of three texture classes, and we found that this ranking varies with the bit rate and the image nature. These results are important because they are the first that take into account texture, one of the basic cues for image recognition.

In future work we intend to make a similar analysis measuring other components of images in order to give a final and more accurate criterion for the selection of an algorithm to compress an image and in order to build a feature preserving image coder. The goal of the future coder will not be to minimize an objective distortion measure at a given bit rate, but to preserve important features of the image. The measure of image quality will be taken with a pattern recognition process.

References

[1] C. Gonzalez Rafael and E. Woods Richard, *Digital Image Processing*, Addison Wesley / Díaz de Santos, U. S. A., 2000.

[2] Saffor A., Bin Ramli R. A., Hoong Ng K. and Dowsett D., Objective and Subjective Evaluation of Compressed Computed Tomography (CT) Images, *The Internet Journal of Radiology*, Vol. 2, No. 2, June 2002.

[3] Yang S., Zhang X. and Mitra S., Performance of Lossy Compression Algorithms from Statistical and Perceptual Metrics, *Proceedings of the 12th IEEE Symposium on Computer Based Medical Systems (CBMS)*, Stamford, CT, USA, pp. 242, June 1999.

[4] Garcia J. A., Fernandez Valdivia J., Rodríguez Sánchez R. and Fernandez Vidal X., Coder Selection for Lossy Compression of Still Images, *Pattern recognition*, Vol. 35, No. 11, pp. 2489 – 2509, November 2002.

[5] Jerabek B., Schneider P. and Uhl A., Comparison of Lossy Image Compression Methods Applied to Photorealistic and Graphical Images Using Public Domain Sources, Technical Report, Research Institute for Software technology, University of Salzburg, April 1998.

[6] Julesz B., Visual pattern discrimination, *IRE transactions on information theory*, vol. 8, pp. 84 - 92, February, 1962.

[7] Lee K. L. and Chen L. H., Unsupervised Texture Segmentation by Determining the Interior of Texture Regions Based on Wavelet Transform, *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 15, No. 8, pp.1231 - 1250, December 2001.

[8] Portilla J. and Simoncelli. E. P., A Parametric Texture Model based on Joint Statistics of Complex Wavelet Coefficients, *International Journal of Computer Vision*, Vol. 40, No. 1, pp. 49 - 71, October 2000.

[9] Brodatz P., *Textures: A photographic album for artists and designers*, New York: Dover, NY, 1996.

[10] Öktem R., Transform Domain Algorithms for Image Compression and Denoising, M. S. Dissertation, Sygnal processing labortory, Tampere University of Technology, Finland, May 2000.

[11] Davis G. and Nosratinia A, Wavelet-Based Image Coding: An Overview, *Applied and Computational Control, Signals, and Circuits*, Vol. 1, No. 1, pp. 205 – 269, 1998.

[12] Shapiro J. M., Embedded Image Coding Using Zerotrees of Wavelet Coefficients, *IEEE Transactions on Signal Processing*, Vol. 41, No. 12, pp. 3445 - 3463, December 1993.

[13] Said A. and Pearlman W., A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees, *IEEE Transactions on Circuits and Systems forVideo Technology*, Vol. 6, No. 3, pp. 243 - 250, June 1996.

[14]. Mrak M., Grgic S. and Grgic M., Picture Quality Measures in Image Compression Systems, *Proceedings of the IEEE International Conference on "Computer as a tool" (EUROCON)*, Vol. 1, Ljubljana, Slovenia, pp. 233 – 236, September 2003.

SOFM Neural Network Texture Segmentation

Erik Cuevas[1,2], Daniel Zaldivar[1,2], Marco A. Pérez-Cisneros[2] and Rául 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, Blvd. Marcelino Garcia Barragan 1421,
44430 Guadalajara, Jalisco, Mexico
marcopc@cucei.udg.mx
(Paper received on August 11, 2006, accepted on September 26, 2006)

Abstract. Texture segmentation is a difficult problem, just as it is apparent from camouflage pictures. A textured region can contain elements of several sizes, each of which can itself be textured. This work presents an algorithm to segment textures. The algorithm is based on self-organizing feature maps which are used to generate a histogram that characterizes the texture. Through classification, the histograms are compared to each other using signal cross-correlation that operates on a defined window according to the texture complexity. The algorithm was tested on benchmark images. The experiments, their results and relevance are presented in the results and future work section.

1. Introduction

Texture is generally recognized as being fundamental to perception. The taxonomy of problems encountered within the context of texture analysis could be that of classification, description, and segmentation. Recognition of texture patterns has applications in radiography and aerial and satellite photography, among others. There is no concise definition or characterization of a texture available in practice. Texture has been described in a variety of ways. Intuitively, texture descriptors provide measures of properties such as smoothness, coarseness, and regularity. One way to describe texture is to consider it as being composed of elements of texture primitives. Texture can also be defined as the mutual relationship among intensity values of neighboring pixels repeated over an area larger than the size of the relationship.

Texture segmentation is the problem of breaking an image into components within which the texture is constant. Texture segmentation involves both representing a texture, and determining the basis on which segment boundaries are to be determined.

Many texture feature extraction and recognition algorithms are available in practice ([1], [2], [3], [4], [5], [6], [7], [8], [9]). Conventional texture recognition algorithms can be grouped into three classes: statistical, structural, and spectral. Statistical approaches yield characterizations of textures as smooth, coarse, grainy, and so forth. Statistical algorithms are based on the relationship between intensity values of pixels; measures include entropy, contrast, and correlation based on the gray level co-occurrence matrix. Structural algorithms are based on image primitives, which they regard as a formational element. Structural algorithms generate, and describe rules for

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 259-268

generating, repeating patterns. The notion of a primitive is central to texture analysis. A *texel* is (loosely) a visual primitive with certain invariant properties. Texels occur repeatedly in different positions, deformations, or orientations inside a given area. Texture primitives may be pixels or aggregates of pixels. One way of describing rules that govern texture is through a grammar. Structural approaches generate patterns by applying the rules of a grammar to a small number of symbols. Spectral techniques are based on properties of the Fourier spectrum and are used primarily to detect global periodicity in the image by identifying high-energy narrow peaks in the spectrum. Both statistical and structural measures lack neurophysiological support [10].

Many neural network models have been suggested for texture recognition ([11], [12]). A generic model for segmenting images by using texture requires the identification of those features that both define texture and allow discrimination between different textures. A class of 2-D filters based on Gabor functions for the texture segmentation was proposed [15], in this approach is shown analytically that applying a properly configured band-pass filter to a textured image produces distinct output discontinuities at texture boundaries. Rao and Vemuri proposed a neural network architecture for texture segmentation and labeling. Their model consists of two major components: the feature extraction network and the texture discrimination network. The feature extraction network is a multilayer hierarchical network governed by Grossberg's boundary counter (BC) system [13]. The texture discrimination network is based on the adaptive learning algorithm devised by Kohonen [14]. Neural network models based on FT-domain feature extraction can also be used for texture feature extraction.

This work this organized in the following way, in the section 2 we analyze the main characteristics of the SOM neural networks and define the used nomenclature, in the section 3 the proposed algorithm is described, finally in the section 4 the results are shown and it is analyzed future work.

2. SOFM neural networks

Self-organizing feature maps (SOFM) [14] learn to classify input vectors according to how they are grouped in the input space. They differ from another networks in that neighboring neurons learn to recognize neighboring sections of the input space. Thus, competitive layers learn both the distributions and topology of the input vectors that they are trained on.

The architecture for a Self-organizing map network is shown in Fig. 1. The |Ndist| box in the figure accepts the input vector **p** and the input weight matrix **IW** and produces a vector having S elements. The elements are the negative of the distances between the input vector **p** and the vector **IW**. The net value **n** of the Self-organizing layer is computed by finding the negative distance between input vector **p** and the weight vector **IW**.

The competitive transfer function C accepts a net value **n** and returns neurons outputs of 0 for all neurons except for the winner, the neuron associated with the most positive element of the input **n**. Thus, the winner's output is 1.

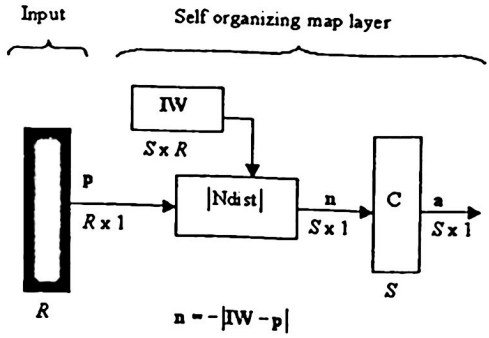


Fig. 1. SOM neural network architecture.

The neurons in the layer of an SOM are arranged originally in physical positions according to a topology function, the more used are square grid, hexagonal and random topology. Distances between neurons are calculated from their positions with a distance function. The Link distance is the most common.

A self-organizing map network identifies a winning neuron i^{th} using the same procedure as employed by a competitive layer, updating not only the winning neuron, but all neurons within a certain neighborhood $N_{i,d}(d)$ of the winning neuron using the Kohonen rule. Specifically, we adjust all such neurons $i \in N_{i,d}(d)$ as follows:

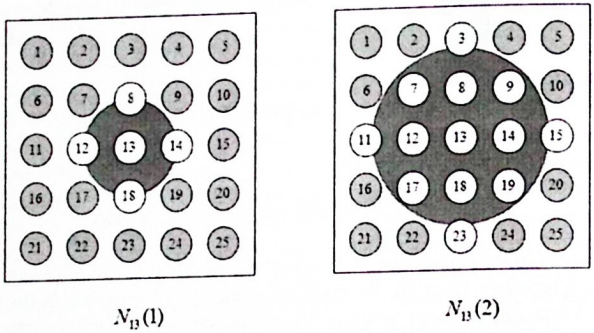
$$_{i}\mathbf{w}(q) = {}_{i}\mathbf{w}(q-1) + \alpha(\mathbf{p}(q) - {}_{i}\mathbf{w}(q-1)) \text{ or} \tag{1}$$

$$_{i}\mathbf{w}(q) = (1-\alpha){}_{i}\mathbf{w}(q-1) + \alpha\mathbf{p}(q) \tag{2}$$

Here α is the learning rate and $N_{i,d}(d)$ contains the indices for all of the neurons that lie within a radius d of the i^{th} winning neuron. Thus, when a vector **p** is presented, the weights of the winning neuron and its closest neighbors move toward **p**. Consequently, after many presentations, neighboring neurons will have learned vectors similar to each other. The winning neuron's weights are altered proportional to the learning rate. The weights of neurons in its neighborhood are altered proportional to half the learning rate. In this work, the learning rate and the neighborhood distance (used to determine which neurons are in the winning neuron's neighborhood) are not altered during training.

Here the *neighborhood* $N_i(d)$ contains the indices for all of the neurons that lie within a radius d of the winning neuron i^{th}. To illustrate the concept of neighborhoods, consider Fig. 2.

$$N_i(d) = \{j, d_{ij} \leq d\} \tag{3}$$



$N_{13}(1)$ $N_{13}(2)$

Fig. 2. At left is shown a one-dimensional neighborhood of radius d=1 around neuron 13, at right is shown a neighborhood of radius d=2.

These neighborhoods could be written as: $N_{13}(1) = \{8,12,13,14,18\}$ and $N_{13}(2) = \{3,7,8,9,11,12,13,14,15,17,18,19,23\}$.

3. Segmentation algorithm

To carry out the segmentation, it is defined a window of p1 x p2 pixels, with the condition that the window contains in "representative form" the structure that the texture defines. With this window the neural net SOM is trained. The neural net architecture has 3 input that represent the x, y coordinates and the pixel intensity while the outputs number depend from the complexity of the structure to segment, although in several experiments were showed to give a good result between 25 and 36 (located in a square arrangement of h x k) neurons. Fig. 3 shows the architecture of the neural net used.

Although the performance of the network is not sensitive to the exact shape of the neighborhoods with the objective of observing the output layer deformation and its covering with the data space of the training window the square grid was chosen, using as distance approach the link distance. The figure 4b shows the deformation presented at the end of the training considering the window shown in Fig. 4(a).

Once trained the net, the net has the capacity to order the input space, that is to say it will learn how to react to repetitive structures found in the window, and considering the training and operation form of these nets, structures that are similar will make to react the neighboring neurons. This is if for an input pattern the neuron 3 reacts as the winner neuron, if a pattern is presented with similar characteristic will react as winner neuron one of the neighboring neurons of 3.

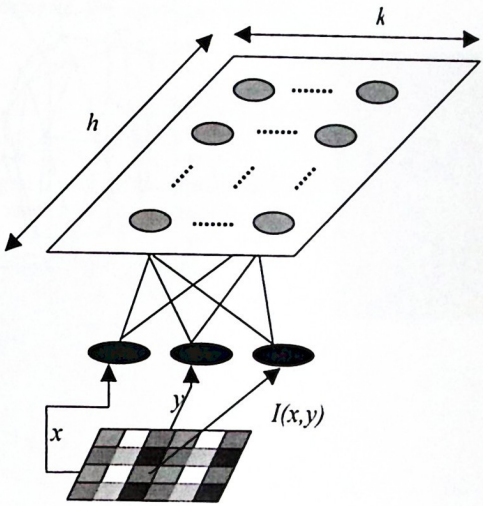


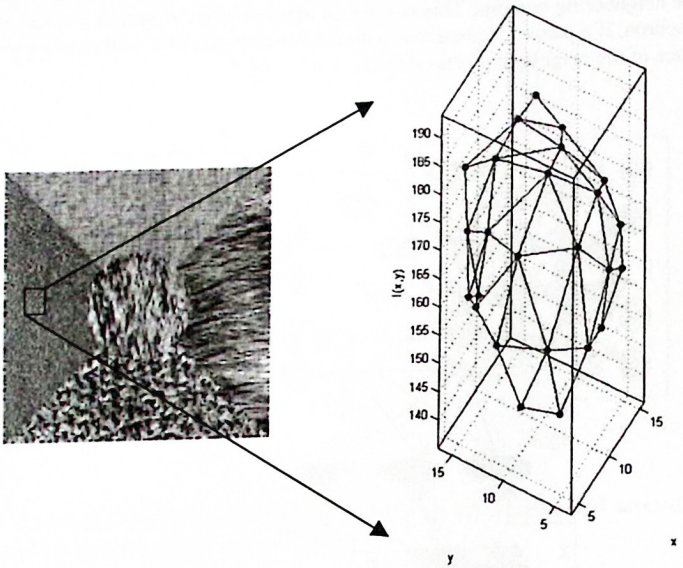
Fig. 3. The neural net architecture has 3 input that represent the x, y coordinates and the pixel intensity.

To characterize the texture using the trained neural network we use the histogram obtained by the net when it is presenting as input the pixels contained in the window. The histogram will contain the information of how many times each neuron was shot as a consequence of have been processed the window pixels. The histogram allows to facilitate the characterization process converting the problem of several dimensions to an one-dimensional problem. Fig. 5(b) shows the obtained histogram considering as training window the figure shown in 5(a).

Considering the histogram as the signal that characterizes the texture we can simply compare this signal with those obtained for the histograms generated by the versions of the window displaced along the image. Many forms exist of comparing these signals however the simple way is to accomplish the cross-correlation of both signals.

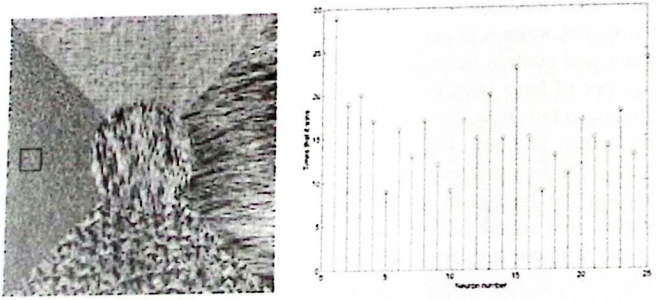
In statistics, the term **cross-correlation** is sometimes used to refer to the covariance between two random vectors X and Y. In signal processing, the **cross-correlation** (or sometimes "cross-covariance") is a measure of similarity of two signals, commonly used to find features in an unknown signal by comparing it to a

known one. It is a function of the relative time between the signals, is sometimes called the *sliding dot product*, and has applications in pattern recognition and cryptanalysis.



(a) (b)

Fig. 4. Figure 4(b) shows the deformation presented at the end of the training considering the window shown in Fig. 4(a).



(a) (b)

Fig. 5. Figure 5(b) shows the obtained histogram considering as training window the figure shown in 5(a).

For discrete functions f_j and g_j the cross-correlation is defined as

$$(f * g)_i \equiv \sum_j f_j^* g_{i,j} \tag{4}$$

where the sum is over the appropriate values of the integer j and an asterisk indicates the complex conjugate. For continuous functions $f(x)$ and $g(x)$ the cross-correlation is defined as

$$(f * g)(x) \equiv \int f^*(t) g(x+t) dt \tag{5}$$

where the integral is over the appropriate values of t. The cross-correlation is similar in nature to the convolution of two functions.

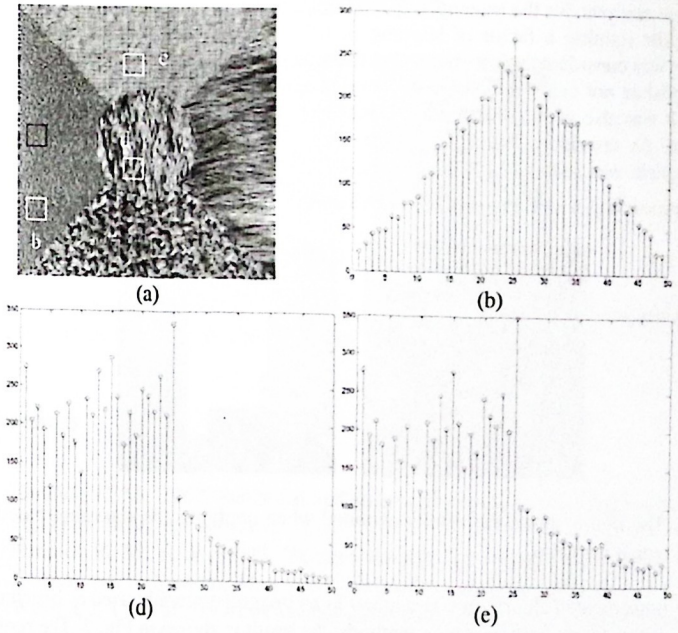


Fig. 6. In the figure 6b is shown the cross-correlation of windows that correspond to the same texture presenting a high correlation among both signals, which does not happen to the signals of the windows histogram of other textures.

In Fig. 6 the histogram obtained in Fig. 5 that characterizes to a texture is compared by cross-correlation with different windows placed in different textures. In Fig. 6(b) it

is shown the cross-correlation of windows that correspond to the same texture presenting a high correlation among both signals, which does not happen to the signals of the windows histogram of other textures.

The segmentation algorithm of the textures is divided in two parts as usually happen in applications of neural networks, they are training and operation. During the training for each texture a neural network SOM is trained. The net architecture is similar to the one shown in Fig. 3. Considering the same training window it is obtained the histogram which will be the signal that characterizes the texture. In the operation the window is displaced by the whole image accomplishing the cross-correlation among the signals that characterizes the texture and the new histogram obtained as a result of applying as input to the net the displaced window.

4. Results and future Work

The algorithm was implemented using the neural architecture SOM proposed, using 25 output neurons, for the training and operation was used a window of 20 x 20 pixels. During the training a factor of learning of 0.2 was used and a neighborhood of 5 neurons was considered, what means that the modifications to the network weights are accomplished not only on the winning neuron, but in those that are in a neighborhood of 5. It was also used in the training a square grid and as distance approach the link distance. As discrimination approach the mean of theobtained cross-correlation of both signals was used being this $\mu_{correlation} = 210$. The training, the correlation and the image processing was implemented in MATLAB.

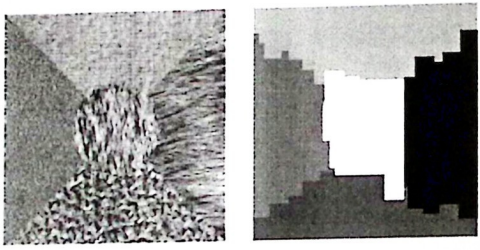


Fig. 7. The figure shows the results obtained when applying the algorithm based on SOFM neural networks.

The implemented algorithm was applied to an image commonly used as benchmark for the comparison of segmentation methods, the result is shown in Fig. 7. The result is good in all homogeneous regions of the textures except in the edges where the algorithm does not allow to discriminate correctly. The Algorithm was proven with other images presented a good performance however the problem in the edge remains constant. The worst case is presented in the edge of several textures types (more of 2) where the window can be classified indistinctly in anyone of them. With the objective of to enlarge the experimental work and to carry out a performance comparison of the

SOFM algorithm, a segmentation texture algorithm is used based on gabor functions such as it is described in [15], both algorithms were applied to Fig. 8 (*Top*), having as results Fig. 8 (*left*), for the algorithm based on gabor functions and Fig. 8 (*right*) for the SOFM algorithm.

The future work can be concentrated on two directions. First the problem of trying with the texture edges using an adaptive method that allows to change the window size and the second is to find a different discrimination method that considers other characteristic (different to cross-correlation) making in this way perhaps a robuster classification.

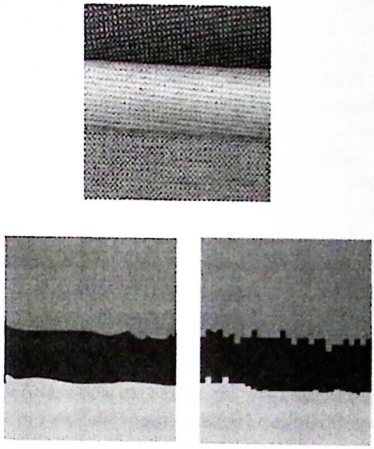


Fig. 8. (*Top*) Original image, (*left*) segmented image using gabor functions and (*right*) segmented image using the SOFM algorithm

References

[1] Haralick, R. M., 1967. Statistical and structural approaches to texture. *Proceedings of the IEEE*, pp. 786-804.

[2] Haralick R. M. et al., 1973. Texture features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, pp. 610-620.

[3] Weszka, J. C. et al., 1976. A comparative study of texture measures for terrain classification. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 6, pp. 269-285.

[4] Ehrich, R. W. and Foith, J. P., 1978. A view of texture topology and texture description. *Computer Graphics and Image Processing*, vol. 8, pp. 174-202.

[5] O'Toole, R. K. and Stark, H., 1980. Comparative study of optical vs. all digital techniques in textural pattern recognition. *Applied Optics*, vol. 19, pp. 2496-2506.

[6] Rosenfeld, A. and Kak, A., 1982. *Digital Image Processing*, vol. I. Orlando, FL: Academic Press.

[7] Ballard, D. H. and Brown, C. M., 1982. *Computer Vision*. Englewood Cliffs, NJ: Prentice Hall.

[8] Coggins, J. M. and Jain, A. K., 1985. A spatial filtering approach to texture analysis. *Pattern Recognition Letters*, vol. 3, pp. 195-203.

[9] Daugman, J. G., 1989. Network for image analysis: Motion and texture. *Proceedings of the International Joint Conference on Neural Networks*, Washington DC, vol. I, pp. 189-193.

[10] Rao, N. and Vemuri, V., 1989. A neural network architecture for texture segmentation and labeling. *Proceedings of the International Joint Conference on Neural Networks*, Washington, DC, vol. I, pp. 127-133.

[11] Clark, M. et al., 1987. Texture segmentation using a class of narrow-band filters. *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, April, pp. 14.6.1-14.6.4.

[12] Kulkarni, A. D. and Byars, P., 1991a. Artificial neural network models for image understanding. *Proceedings of the SPIE Conference on Image Processing Algorithms andTechniques II*, San Jose, CA, vol. 1452, pp. 512-522.

[13] Grossberg, S. and Mingolla, E., 1988. A neural network architecture for preattentive vision: Multiple scale segmentation and regularization. *Proceedings of the International Joint Conference on Neural Networks*, San Diego, pp. 177-184.

[14] Kohonen, T., 1988. *Self-Organization and Associative Memory*. Berlin: Springer-Verlag.

[15] D. Duna, Higgins W. E. and Wakeley J., Texture Segmentation using 2-D Gabor Elementary Functions, *IEEE, Tansactions on pattern análisis and machine intelligence*, February 1994 (Vol. 16, No. 2) pp. 130-149.

License Plate Location and Recognition Using Decision Regions of Color with Genetic Algorithms and Neural Networks

Gilberto Arístides Apodaca Aragón and Patricia Rayón Villela
Tecnológico de Monterrey, Campus Ciudad de México, Calle del Puente No.222. Col Ejidos de Huipulco, 14380, Tlalpna, México D.F. México
prayon@itesm.mx

(Paper received on August 17, 2006, accepted on September 27, 2006)

Abstract. Automatic license plate location and recognition is an important field of opportunity for law enforcing. Cars involved in crimes can be located a lot faster by a spread number of cameras looking for them automatically than by visual inspection by police officers. This paper proposes a method for license plate location and recognition of moving vehicles with complex background. The license plate is located using color decision regions, with a two stage process using genetic algorithms and a multilayer perceptron neural network; finally individual characters are segmented using color and then a neural network is used to classify them.

1. Introduction

License plate recognition is an important research topic in computer vision nowadays. The applications range from traffic control, authentication to enter private buildings; to toll systems in paying parking lots.

The license plate recognition problem is always divided in two steps: license plate location and character recognition. For the location process, techniques such as vertical edge detection [12], color classification using a neural network [9], vector quantization [15], fuzzy logic [13], as well as directly looking for sequences of characters in the source image are used.

The recognition problem is usually further divided into two stages [2]: character segmentation and character recognition. Techniques often used for character segmentation include discrete time cellular network (DTCNN) [1], horizontal and vertical histogram accumulation through the slopes of vertical dynamic range [10] and horizontal and vertical component connection count. For character identification Jaccard value based templates [9], skeletization, intersection location and line tracing [3] and neural networks [2], among others were used.

Different approaches in literature, especially for license plate location, are not directly comparable due to differences in license plates in various locations throughout the globe and the different environments in which license plates are acquired.

Pan et al [6] use black, wide license plates (aspect ratio of about 4:1) with a thin white frame and white letters. Lim et al [3] use the same kind of license plates: Black, wide with a white frame. Their paper also assumes the license plate has been pre-

© H. Sossa and R. Barrón (Eds.)
Special Issue in Neural Networks and Associative Memories
Research in Computing Science 21, 2006, pp. 269-279

segmented, their sample image shows the license plate up close with only some of the back of the car side showing. Zheng [12] recognizes the same kind of license plates. This paper does cover location and environmental noise. Brugge [1] uses a wide license plate with yellow, reflective and black letters in pairs. The sample images contain background information, however they are taken from a fixed location and therefore the background is static. Chang [2] reads white license plates with black numbers that have an aspect ratio of about 2:1. Chang uses a variety of angles and complex, real life backgrounds. Nijhuis [5] reads license plates with reflective yellow backgrounds, black numbers, wide aspect and a black thin border; also uses images where the car fits entirely in the scene but almost fills it, not leaving much background in the scene. Park [7] uses black tall license plates with an aspect ratio of about 2:1, white letters and a white border. Images where the car almost completely fills the image as well are used. Ryung [9] works with the same type of license plates: black, white letters, white border and 2:1 aspect ratio. The test images have a static background. Sirithinaphong [11] uses 2:1 license plates with white background and black letters without much environmental noise.

This paper proposes different hybrid methods for license plate location, and recognition with complex background. First, a license plate location method using vertical edge detection and aspect ratio information was tested. A second one that uses two different neural networks, one to perform color segmentation and the other one to identify license plate candidates. Finally, the third one uses a genetic algorithm to find Decision Regions of Color (DRC) in the RGB color space to perform color segmentation and then a neural network to classify this DRC is used. Then the recognition stage is done using a neural network.

The results of this paper, and other papers in literature, can not be directly compared because of the differences in the license plates as well as the differences of the environments under which the images of the plates are captured. Furthermore, some papers consider location or recognition of the plate only, not both.

2. Methodology

The proposed methodology is divided in four stages: preprocessing, license plate detection, character segmentation and character recognition. In the first two stages, it can be considered that the image may contain zero, one or more license plates, with natural (complex) backgrounds.

Three license plate detectors were proposed. The first license plate detector looks for vertical lines in the preprocessed image and then uses the aspect ratio information of the lines found to figure out where the plate may be; the second detector uses a neural network to convert the colors in the original image into predefined color classes; separating what is a license plate from what is not. Finally, for the third detector, a genetic algorithm was used to find a Decision Region of Color (DRC) in the RGB space that define the colors contained in the plates and finally the image was then segmented using those colors with a multilayer neural network perceptron.

The character segmentation module takes a license plate image and separates the

characters using color information from the original image, together with an heuristic function to construct the complete characters with the lowest amount of noise.

The character identification module extracts a set of features from the segmented characters and uses them as input for a multilayer perceptron neural network for classification.



Fig. 1. This figure illustrates vertical edge detection. **A)** Original image **B)** Sobel operator **C)** Modified Sobel operator

3. Preprocessing

The contrast in the license plates that were being located is very low, so the images were first equalized. Then the image is converted to gray-scale, and then the vertical edges needed for locating the plates were extracted using a custom convolution kernel based on the Sobel operator. The known Sobel operator's convolution kernel was modified so that it smooths the image, and (as a side effect) slightly thickens the borders.

Due to the normalization of the image, an iterative binarization method based on two thresholds and 3 regions is proposed; the algorithm is a modified version of the iterative threshold selection method.

First an initial threshold value is obtained by the gray level image average, this value is used to divide the image in two different regions, further two different thresholds for each region are obtained, and also 3 different regions. This procedure is repeated until the threshold values do not change.

Finally, a simple noise reduction algorithm is applied to the binarized image to remove small and irrelevant objects that are generated by complex environmental conditions, by sliding a window and deleting any object that completely fits in it.

4. Proposed methods for license plate location

Three different methods for license plate detection were used to compare their performance. The first one is based on vertical edges detection, and the other two are color-based methods.

4.1. Vertical edges based method

This method let us extract candidate license plates from the noise reduced image, it consists in finding matching pairs of vertical lines with approximately the same height, y coordinates and separated by an aspect ratio of approximately 2:1. A variety of factors, including vanity plate holders, crooked license plates and noise in the acquisition stage, force to have very loose parameters for each step in this process, resulting in a high number of false positives, so this method is not described in detail in this paper.

4.2. Color based methods

These methods are based on the color of the license plates, and the DRC defined by that color in the RBG model. The image is color segmented according to colors that are known to belong to license plates, and colors that are known not to. Each license plate colored region is analyzed in order to extract features that describe its DRC. These features are classified by a neural network in order to determine whether they describe a license plate or not.

Two approaches were tried in constructing the DRC in the RGB color space. The first one built by hand and classified by a neural network; and the second one built automatically using a genetic algorithm and classified using a neural network.

4.2.1. Neural network based color segmentation

For the neural network based color segmentation, a hand crafted set of license plate colors was used, then a set of no license plate colors was generated automatically and both were used to train a neural network to classify both. This method is intended to identify license plates under the broadest possible spectrum of light sources and conditions; therefore the images were taken under direct and indirect sunlight, incandescent and fluorescent lights sources. The color set was obtained using 63 different images containing the rear license plate. The distance of the cars from the camera ranged from 1 to 7 meters. The lens' zoom was set at its furthest position.

License plates were extracted by hand from these images and collected together in a single mosaic image. Special attention was taken so that the plates were cut containing no regions of the image that don't strictly belong to the license plates; which means screws, vanity license plate holders and other objects were removed from the image even if they overlapped with the plate.

The number of colors in the image was then reduced from 24-bit color to 8-bit (255) and from there, colors belonging to the foreground (letters) and the background of the license plate were selected manually from the image's palette.

To generate the non license plate colors, the previous set was converted into a DRC in the RGB color space. The region is described by its vertices which are defined by the colors in the license plate background color set.

To construct the DRC, each of the colors in the set was tested against the rest. Every one of them that was not entirely contained by any other 4 colors in the set became a

vertex of the region.

A big sample of colors was generated, and each of those tested for inclusion in the DRC. Any sample that was not completely included in the DRC or touched its borders was added to the non license plate colors set; if the sample was contained in the DRC, it was added to the enhanced license plate background color set; if the sample touched the borders of the shape, it was discarded because it was left up to the neural network to finely define the borders of the shape, and such samples would have over-trained it.

The non license plate color, enhanced license plate background color as well as the license plate foreground color sets were used as input to train a back propagation neural network with one hidden layer, 3 input nodes, 3 output nodes and 128 nodes in the hidden layer. The number of nodes in the hidden layer was determined experimentally. It was believed to be too big, but that configuration provided best results.

The neural network is then used to classify colors in source images to create regions of similar colors. Any region that belongs to the license plate background color set is considered to be a license plate candidate.



Fig. 2. This figure shows the license plate located and highlighted in white. A) Original images. B) Segmented images.

4.2.2 Genetic algorithms and neural network color segmentation

For the geometry based color segmentation, four *(x,y,z)* coordinates, that define a Decision Region of Color (DRC), were used to define the license plate background color set. The set was later fine tuned by adding intersecting regions that subtracted colors from the license plate background color set, adding them to the non-license plate color set.

In order to generate the DRC, a training set of 100 pictures of cars was taken from behind, under similar lighting conditions, with a range of distances from 5 to 8 meters

and with the cars being at angles between 30 and 60 degrees relative to the camera. The pictures were taken under direct sunlight, between 11:00 and 13:0 during the spring.

The license plates were manually extracted from the images. Special attention was paid to making the cut leaving a buffer zone around the plates that could include non license plate colors; therefore the license plates themselves must never touch the edges of the cut images.

A genetic algorithm was then used to generate the DRC that would be used to classify the colors in the test images. The goal of the genetic algorithm is to find a configuration of DRC that is capable of finding the license plate in the biggest number of images possible while, simultaneously, making sure the license plate regions do not extend beyond their borders, invading zones of the image surrounding the plate (typically the trunk of the car).

Fitness function. In order to achieve this configuration, a fitness function was developed so that the algorithm can find a configuration by performing the following optimizations:

1) Maximize the size of the biggest region in each image. The biggest object should always be the license plate background, including the "angel" image in it. It may or not include the letters, but that isn't as important as they are separated in another stage in the process; even though it is still better if it doesn't because it means the color set fits the background colors more tightly, and will generate less false positives.
2) Minimize the amount of pixels of the biggest region that touch the borders of the image. That means that the license plate should be completely contained within the image and not bleed out, or at least be contained as much as possible. It is important to remark that this step doesn't imply minimizing the number of pixels of the license plate background set that touch the borders of the image; that is still possible especially for golden cars; the goal of this step is that the one object that is assumed to be the license plate should be separated from other objects that touch the border; despite the color set to which those belong.
3) Generate a small number of regions. So that both halves of the plate, and the "angel" (that divides those two halves of the plate) are covered by the color set. A potential side effect of this restriction is that the letters of the plate may be included in the wrong set; but as mentioned before that is not a big problem.

Encoding. The red, green and blue components of each color that define the vertices of the DRC were stored as real (floating point) numbers to have more precision when the colors are separated; especially considering that each solid is only defined by 4 points, and that makes it hard to include colors in the DRC when they're away from its center of gravity because the angles become too acute.

By using floating point numbers, and ranges that extend the intensity ranges of the image ([0.255]) more colors are possible to be integrated inside the sets.

Range and precision: The range of intensity values of each channel of the image is [0.255]. The DRC vertices are allowed to exceed these borders because of the

limitations of the representation. Therefore, the range of values per coordinate per pixel that the genetic algorithm seeks was extended to (-256,512).

In order not to waste most of the generated individuals, the range is expanded in a zigzagging fashion. That is, after the bounds are exceeded, the direction of the numbers changes until it meets the opposite bound again. For instance, the number 513 is converted into 512 and so on.

The resulting DRC are then used to classify the colors in the source images as license plate or non license plate colors. With those, the DRC are grouped and later classified by a neural network as license plates or not, as mentioned above.

In order to determine whether a color is contained by one of the DRC, a plane is formed with 3 colors of the solid, then it is determined whether the remaining color is bigger or smaller than the plane using the straight line general equation; finally, it is also determined whether the color that is being evaluated is bigger or smaller than the plane. If the remaining color was bigger and the color being evaluated is smaller, or vice versa; the color is considered to be outside of the DRC; otherwise, the color is potentially included. The previous steps are repeated for all 4 possible combinations of plane - stray point colors. If the color is considered potentially inside for all combinations, then the color is determined to be contained in the DRC; otherwise it is not.

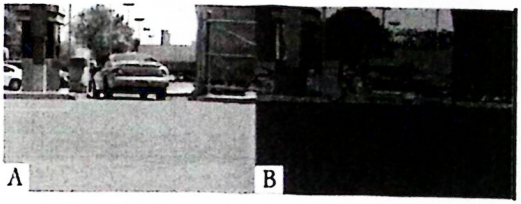


Fig. 3. This figure shows a failed color segmentation. **A)** Original image **B)** Color segmented image

Figure 2 shows an example of correct color segmentation of the license plate in three images under different lighting conditions and real life backgrounds. Figure 3 shows an example of an incorrect segmentation, where the color of the car is too similar to the color of the plate for it to be segmented.

5. Proposed method for license plate recognition

5.1. Character Segmentation

In this paper, a color-based image segmentation technique was used to separate the characters. Sixty sample hand-cut license plate images, taken in diverse lighting conditions were examined to generate a color table with the following categories: background colors, foreground colors, mostly background colors and mostly

foreground colors.

The total length of the table became 156 colors; the table was further optimized by finding colors that are contained by sets of other colors in RGB space and eliminating them from the table.

In order to classify the pixels, the license plate images are analyzed pixel by pixel, finding the color in the table closest to the color in a pixel and then labeling it with the corresponding tag from the table. This generates an image with 4 different values, as shown in Figure 4.

In order to assign ambiguous regions to the foreground or the background, a number of combinations are tried and analyzed with a heuristic function; the regions are combined as follows, for every character found: (a) foreground is foreground and everything else is background. (b) foreground and "mostly foreground" are foreground and everything else is background. (c) background is background and everything else is foreground and (d) a number of randomly (probabilistically) generated combinations. The one with the best heuristic value is chosen as a character.



Fig. 4. This figure shows character segmentation. **A)** Original image **B)** Segmented image.

b) Feature extraction

Each character is first normalized to a fixed size and adjusted aspect ratio using continuous aspect ratio mapping [4], which was proved to be better than proportionally scaling the characters. The equation for continuous aspect ratio mapping can be seen in Equation 1, where r is the original image's aspect ratio and r' is the resulting image's aspect ratio.

$$r' = \left| \sin\left(\frac{\pi}{2} r\right) \right| . \tag{1}$$

A reduced Hough transform is used to identify horizontal, vertical, 45 degrees and 135 degrees lines. The three most populated lines are selected, and their origin, angle and number of votes are used as features, after sorting the lines to assure they are

always used in the same order. The normalized distance of every white pixel from the center is calculated and used as another feature; and finally, the image is divided in 6 regions (in 2 horizontally by 3 vertically), and the distance from the points located at the center and edges of several of those regions to white pixels within those regions are calculated. The points are shown in Figure 5.

c) Character identification

For number identification a 2 layer, back propagation neural network was used with

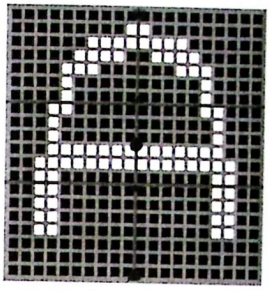


Fig. 5: Figure of feature points from which distance is measured

20 input nodes (corresponding to the 20 features extracted); 30 nodes in both hidden layers and 10 output layers.

6. Results

The vertical edges based method for license plate location had a success rate of 80% on the set of training images; however, the success dropped below 60% on test images due mostly to low contrast between the plates and the cars and it being overly sensitive to deformations on the edges of the license plate, often seen in Mexico city.

The color segmentation using a neural network had a success rate of 85% both on the training and test sets.

The performance of the neural network in the license plate recognition was improved when the genetic algorithm for DRC in the RGB color space was used, with success rate of 95% on the training images set, and 92% on the test set.

Both color-based methods are highly successful at finding plates when vanity license plate holders are installed, with success rates of over 99%.

Both methods have problems with certain kind of colors. The most problematic is golden, because it is similar to the color of the plate; under some lighting conditions it may appear identical to how the plate appears under other lighting conditions; thus, the plate is near impossible to extract from the car. Other problematic car colors are white

and silver.

The recognition method had a success rate of 98% on numbers and 97% on letters. The character segmentation's performance can't be evaluated individually, because characters that may appear wrong visually were still recognized properly by the neural network.

Of the literature, the method presented by Zheng [12] can not be applied, and therefore fairly compared to the one presented in this paper because the complex backgrounds considered in this one would yield a high number of false positives with Zheng's edge counting method; Zunino [15] assumes the distance and position of the plate to be known, conditions that are not met in this paper. Ryung [9] utilizes a neural network to perform color segmentation of the source images, similar to the article, together with horizontal and vertical histogram count, the method isn't comparable to the presented one in this paper because it uses a pixel's neighbors' to perform the color segmentation, but because the size of the plate may be very big in this paper, if the plate is located near the camera, the pixel's neighbors may all be of one color, whereas Ryung's method depends on the neighbors having both background and foreground license late colors; finally, Lim's method looks for contiguous regions to look for characters directly on the image, which would be impossible to do under the environment of this paper because the contrast of the colors of the plate isn't enough to directly separate the background from the foreground without knowing the location of the characters first.

References

[1] Brugge M, Stevens J, Nijhuis J, Spaanenburg J. (1998). License Plate Recognition using DTCNNs. Fifth IEEE International Workshop on Cellular Neural Networks and their Applications.

[2] Chang S, Chen L, Chung Y, Chen S. (2004). Automatic License Plate Recognition. IEEE Transactions on Intelligent Transportations Systems.

[3] Lim B, Yeo W, Tan K, Teo C. (1998). A Novel DSP Based Real-Time Character Classification and Recognition Algorithm for Car Plate Detection and Recognition. Proceedings of ICSP.

[4] Lin-Liu Ch, Nakashima K, Fujisawa H. (2003). Handwritten Digit Recognition: Benchmarking of State of the Art Techniques. Pattern Recognition.

[5] Nijhuis J, Brugge M, Helmholt K, Plium J, Spaanenburg L. (1995). Car License Plate Recognition with Neural Networks and Fuzzy Logic. Proceedings, IEEE international Conference on Neural Networks.

[6] Pan X, Ye X, Zhang S. (2005). A Hybrid Method for Robust car Plate Character Recognition. Engineering Applications of Artificial Intelligence.

[7] Park S H, Kim K I, Jung K, Kim H J. (1999). Locating Car License Plates using Neural Networks. Electronics Letters Online.

[8] Reichenbach, Stephen E.; Park, Stephen K. (1990). Alter-Gartenberg, Rachel.: Optimal Small Kernels for Edge Detection, 1990. Proceedings. 10th International Conference on Pattern Recognition.

[9] Ryung E, Pyeoung K, Joon H. (1994). Automatic recognition of a car license plate using color image processing. IEEE International Conference.

[10] Salgado L, Menéndez J, Rendón E, García N. (1999). Automatic car plate detection and recognition through intelligent vision engineering. Proceedings 33rd annual 1999 international Carnahan conference on Security Technology.
[11] Sirithinaphong T, Chamongthai K. (1999). The recognition of car license plate for automatic parking system. Fifth International Symposium on Signal Processing and its Applications.
[12] Zheng D, Zhao Y, Wang J. (2004). An Efficient Method on License Plate Location. Pattern Recognition Letters.
[13] Zimic N, Fickzo J, Mraz M, Virant J.: The Fuzzy Logic Approach to the Car Number Plate Locating Problem. Intelligent Information Systems, Proceedings.
[14] Ziou, Djemel; Tabbone, Salvatore (1998). Edge Detection Techniques – An Overview. Technical report, No. 195. Dept Math & Informatique, Universit de Scherbrooke.
[15] Zunino R, Rovetta (2000). Vector Quantization for License Plate Location and Image Coding. IEEE Transactions on Industrial Electronics.

Author Index
Índice de autores

Aguilar, Nadezhda 117
Akimitsu, Toshio 105
Andina, D. 97
Apodaca, Gilberto A. 269
Barouh, Ieroham S. 131, 151
Barrón, J. Miguel 97
Barrón, Ricardo 49, 227
Britos, Paola 195, 205
Cabrera, Agustín 25, 177
Capistran, Marcos 161
Cataldi, Zulma 205
Chairez, Isaac 13, 25, 117
Cisneros, Hermenegildo 227
Cortina, Maria G. 97
Cruz, Benjamín 49, 227
Cuevas, Erik 141, 259
De la Cruz, Irving P. 131
Dominguez, David 39
Dzuga, Grzegorz 173
Escalante, Saul 151
Farguell, Enric 3
García, Ariel L. 217, 237
Garcia, Christophe 61
Garcia, Ramon 195, 205
Gómez, Eduardo 3
González, Miguel 217, 237
Grosser, Hernan 195
Hernández, Neil 217, 237
Hirose, Akira 105
Ibarra, Rodolfo E. 217, 237
Koroutchev, Kostadin 39
Lalit, M Patnaik 87
Lara, Bruno 161
Ledesma, Sergio 97
Lefebvre, Grégoire 61
Mariaca, Carlos R. 131, 151
Mazzanti, Ferran 3
Méndez, Luis A. 75
Miranda, Ramón F. 25
Mora, Jaime 217, 237
Okabe, Yoichi 105
Perec, Andrzej 173
Pérez, José H. 181
Pérez, Marco A. 141, 259
Pinto, Raúl 249
Poznyak, Alexander 13, 181
Poznyak, Tatyana 13
Rendón, Juan M. 161
Reyes, Orión F. 75
Ruíz, José 249
Rodríguez, Francisco 39
Rayón, Patricia 269
Rojas, Raul 141, 259
Ruzylo, Mateusz 173
Salgueiro, Fernando 205
Serrano, Eduardo 39
Sierra, Enrique 195, 205
Sossa, Humberto 49
Srinivas, Kumar M 87
Srinivasa, K G 87
Vega, A. 97
Venugopal, K R 87
Vergara, Osslan O. 249
Vinay, T C 87
Zaldivar, Daniel 141, 259

Editorial Board of the Volume

Comité editorial del volumen

Alejandro Rosete
Alessandro E.P. Villa
Alexander Poznyak
Alfredo Weitzenfeld
Andre Ponce de Leon
Ángel Kuri
Antonio Bahamonde
Arturo Hernández
Bart Kosko
Bernard Widrow
Bert Kappen
Carkis A. Reyes
Changyin Sun
Changjiu Zhou
Chilukuri K. Mohan
Christian Jutten
Christophe García
Danil Prokhorov
David Hoyle
Derek Partridge
Du Zhang
Edgar Sánchez
Eduardo Gómez
Exiquio Leyva
Francisco Cuevas
Francesco Masulli
Francesco Morabito
Franz Wotawa
Georgios Anagnostopoulos
Giacomo Indiveri
Gustavo Arroyo
Héctor Pérez
Huaguang Zhang
Ieroham S. Baruch
Janusz Kacprzyk
Jesús Favela
Jesus Cid-Sueiro
José A. Ruz Hernández
Wlodzislaw Duch
Xiao Hui Liu
Jesús G. Figueroa
John Qiang Gan
José Dorronsoro
Juan Flores
Juan Pedro Febles
Klaus Obermayer
Khurshid Ahmad
Lipo Wang
Luiza de Macedo
Marcilio C. P. De Souto
Manuel Graña
Manuel Mejía Lavalle
Marley Vellasco
Marcos Eduaro Valle
Miguel Á. Garay Garcell
Nikolaos Bourbakis
Nikola Kasabov
Oscar Castillo
Okyay Kaynak
Patricia Melin
Patricia Rayón
Pedro Y. Piñero Pérez
Peter Andras
Péter Érdi
Peter Sussner
Peter Szolgay
Raúl Gardúño Ramírez
Raúl Rojas
Richard Duro
Rogelio Soto Sanqing Hu
Soo-Young Lee
Stan Gielen
Teresa Ludermir
Witold Pedrycz

Impreso en los Talleres Gráficos
de la Dirección de Publicaciones
del Instituto Politécnico Nacional
Tresguerras 27, Centro Histórico, México, D.F.
Noviembre de 2006.
Printing 500 / Edición 500 ejemplares.

Neural Networks, Associative Memories together with Fuzzy Systems and Genetic Algorithms are part of the 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 25 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:

- Algorithms and Learning
- Associative Memories
- Data Analysis, Pattern Recognition and Self-Organization
- Neural Control and Robotic Applications
- Real World Applications
- Support Vector Machines
- Vision and Image Processing

ISSN: 1870-4069
www.cic.ipn.mx/rcs
www.ipn.mx

70 Aniversario
1936 • 2006

INSTITUTO POLITÉCNICO NACIONAL
"La Técnica al Servicio de la Patria"