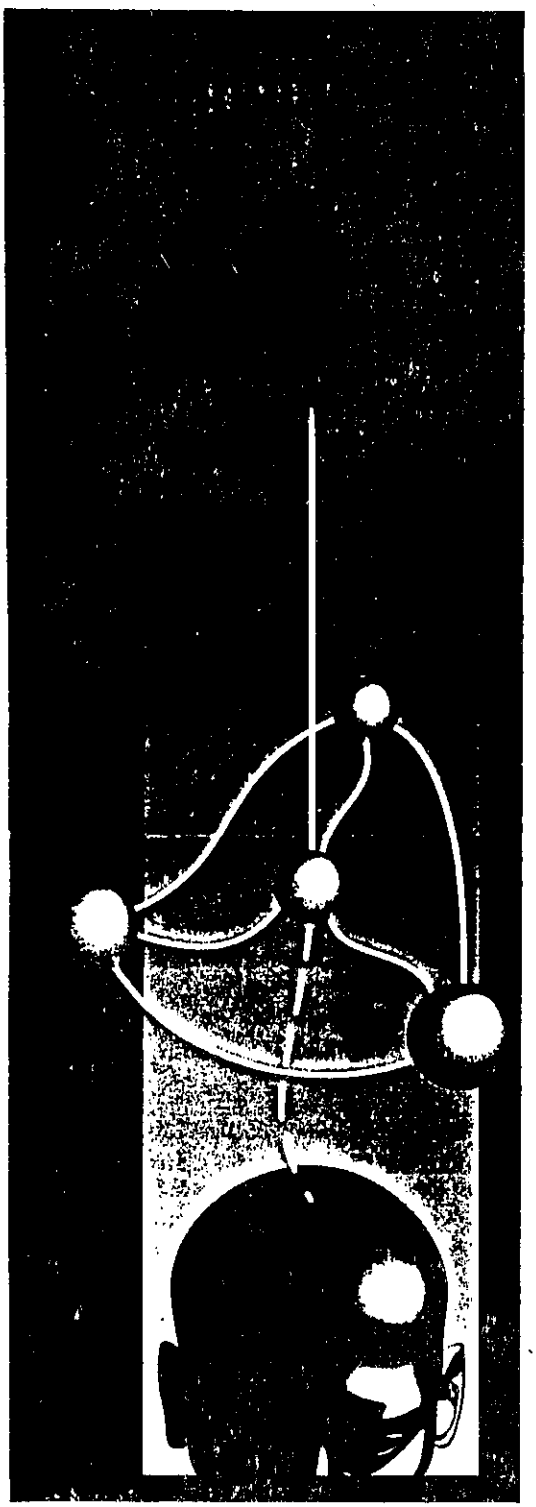


Anais do 1° Congresso Brasileiro de Redes Neurais

Itajubá, 24 a 27 de outubro de 1994.

Presidente do Conselho:
Carlos Eduardo Pedreira

Presidente da Comissão Organizadora:
Germano Lambert Torres

Presidente do Comitê de Programa:
Alexandre Pinto Alves da Silva

Promoção:
Conselho Nacional de Redes Neurais

Organização:
Escola Federal de Engenharia de Itajubá

Anotações

Palavras do Presidente do Conselho Nacional de Redes Neurais

No primeiro semestre de 1992 surgiu a idéia de se organizar um evento em Redes Neuronais no Rio de Janeiro. Em outubro deste mesmo ano se realiza na PUC-Rio o 1° Workshop Nacional de Redes Neuronais em conjunto com a 1° Escola de Redes Neuronais. Os meses que antecederam esta realização foram de muito trabalho para meia dúzia de entusiastas que acreditaram na idéia, e o resultado se mostrou compensador. Reuniram-se por quatro dias cerca de 160 participantes, incluindo a destacada participação dos Professores Manoel Tenório e Igor Aleksander, que vieram dos Estados Unidos e da Inglaterra especialmente para o evento. Foram palestras diversas, mesas redondas e muito bate-papo, talvez a parte mais importante. Pela primeira vez pesquisadores ligados a Redes Neurais se reuniam no Brasil para tomar conhecimento do que estava sendo trabalhado individualmente e para conversar sobre como podíamos colaborar para impulsionar esta área no país. Começamos por polemizar com o próprio nome, que por decisão da maioria passou a ser Neurais e não Neuronais. Uma das questões centrais era como tratar esta área que não é exatamente Computação, nem Física, nem tão pouco Automação ou Biologia, mas uma rica mistura de tudo isso e mais alguma coisa. Discutiu-se a possibilidade da criação de uma Sociedade Brasileira de Redes Neurais mas acabou prevalecendo a opinião, quase de consenso, de que seria melhor aguardar um maior amadurecimento da área, e a ampliação da discussão possibilitando a participação dos que não puderam estar presentes neste primeiro encontro. Surge aí o Conselho Nacional de Redes Neurais, primeiramente em versão provisória e cerca de um ano depois na forma definitiva.

Passados dois anos, partimos agora para uma segunda etapa marcada pela realização do 1° Congresso Brasileiro de Redes Neurais. Muito nos alegra a presença de quatro convidados vindos do exterior. Estarão entre nós Yaser Abu-Mostafa, Yoh-Han Pao, Dejan Sobajic além do Tenório que nos honrará mais uma vez com a sua participação. Muita coisa evoluiu desde o Workshop mostrando alguns sinais de amadurecimento da área no Brasil. Estamos publicando mais de 50 artigos selecionados entre os recebidos de muitos pontos do país e do exterior, representando de forma abrangente a produção científica em Redes Neurais no Brasil.

Em nome do Conselho Nacional de Redes Neurais desejo agradecer a todos que colaboraram na viabilização deste primeiro congresso que esperamos seja o início de uma longa série.

Aos que participam nossas ótimas vindas !

Carlos Eduardo Pedreira
Presidente do Conselho Nacional de Redes Neurais

Palavras do Presidente do Comitê de Programa

O trabalho do Comitê de Programa em revisar os artigos de um Congresso nem sempre é uma tarefa fácil, pois normalmente somos confrontados com situações "sui-generis". Uma nova aplicação ou um tratamento informal ao problema podem gerar dificuldades para a revisão. Neste Congresso, este fato só vem a se agravar pois diversas áreas do conhecimento, das mais distintas possíveis (como visão e previsão hidrológica, ou, sistemas cardiovasculares e jogo de damas), concorrem para a participação no evento.

Para este julgamento foi necessário constituir um Comitê bastante eclético e que abrangesse a maior parte das áreas do conhecimento. A tarefa deste Comitê foi árdua e contou com o apoio de diversos outros profissionais que anonimamente contribuiram para este trabalho. A eles o nosso mais sincero agradecimento.

Finalmente, chegou-se aos 59 artigos que foram solicitados pela Comissão Organizadora, divididos em 15 sessões técnicas (4 em cada uma delas e 3 na de Neurociência e Biomédica). Tentou-se dividir os artigos em dois grupos: desenvolvimentos teóricos e aplicações.

Nos desenvolvimentos teóricos foram criadas sessões de Fundamentos, Novas Arquiteturas, Algoritmos de Treinamento, Heurísticas e Ambientes de Desenvolvimento. Na parte de aplicações foram criadas sessões de Processamento de Sinais I e II, Processamentos de Imagens, Visão Computacional, Controle I e II, Sistemas Elétricos de Potência, e Neurociência e Biomédica.

Após esta classificação, alguns artigos não puderam segundo sua natureza serem distribuídos na sessões acima citadas. Para eles foram criadas duas sessões denominadas Miscelâneas I e II, que com certeza prenderão a atenção dos participantes devido a seu aspecto multi-disciplinar.

Devo agradecer finalmente aos membros do Comitê de Programa que realizaram um excelente trabalho de revisão e classificação dos artigos, muitas vezes mesmo em seus períodos de férias. Um trabalho árduo mas certamente compensador. A todos o meu muito obrigado!

Alexandre Pinto Alves da Silva
Presidente do Comitê de Programa

Palavras do Presidente do Comissão Organizadora

Quando recebi a incumbência de organizar o 1° Congresso Brasileiro de Redes Neurais e, em conjunto, a 2° Escola de Redes Neurais tinha a certeza do desafio que iria encontrar. Primeiramente, sendo realizado o evento em uma pequena cidade universitária, Itajubá, deveria compensar os problemas de translado com o fraternal jeito de ser do mineiro e com as riquezas da terra. Segundo, pois o 1° Congresso deveria ser efetivamente o primeiro e não o único, com isto seu sucesso seria imprescindível para a realização dos demais. A terceira dificuldade seria os parcos recursos disponíveis para a organização de eventos e de apoio aos pesquisadores e palestrantes, fato que seria agravado com o número de eventos técnico-científicos em curso neste segundo semestre.

Porém nada disto seria um impecílio intransponível, com trabalho e imaginação todos eles, um a um, poderiam ser ultrapassados com sucesso. Para tal, foram formadas duas comissões de apoio: a Organizadora e a Local.

A Comissão Organizadora teve como missão coordenar a busca de auxílios financeiros, a divulgação do evento em nível nacional e, principalmente, organizar as bases para o Congresso e a Escola.

O Comitê Local formado por professores e funcionários da EFEI tratou de coordenar as atividades locais, de infra-estrutura e sociais.

Devo deixar meus agradecimentos aos membros destas comissões e dizer que sem eles não seria possível ultrapassar todas as barreiras.

Agradecimentos também devem ser dados às companhias e orgãos de fomento que em muito nos auxiliaram, notadamente, entre outras: CEMIG, IBM, American Airlines, GEC Alsthom, FAPEMIG, FAPERJ e CNPq. A eles nosso preito de gratidão.

Gostaria também de agradecer a direção da Escola Federal de Engenharia de Itajubá e aos meus colegas que ouvidaram todos os esforços para que este evento fosse um sucesso.

Finalmente, uma palavra de agradecimento aos autores, figuras principais do evento, que proporcionaram trabalhos de alta qualidade técnica. Sabe-se das dificuldades em realizar pesquisa em um país sem muitos recursos, mas com nossa criatividade conseguimos levar nossos trabalhos além das fronteiras do país.

O 1° CBRN e a 2° ERN foram organizados para vocês congressistas,

sejam bem-vindos à EFEI e à Itajubá !

Germano Lambert Torres
Presidente da Comissão Organizadora

Uma História de Realizações ...

Europa, 1913, o mundo está as vésperas do mais sangrento conflito bélico já registrado: a 1° Guerra Mundial. A Bélgica se diz neutra mas se encontra entre França e Alemanha, os dois pólos do conflito.

Brasil, 1913, vigora a política do "café-com-leite". Era Presidente da República, o Marechal Hermes da Fonseca e o seu Vice, Wenceslau Braz Pereira Gomes, que representava o lado mineiro desta política. Dr. Wenceslau era filho e genro de poderosos coronéis no sul de Minas.

É dentro deste nebuloso cenário mundial que o jovem Theodomiro Carneiro Santiago, cunhado de Wenceslau Braz, sai em missão aos Estados Unidos da América e à Europa, com recursos próprios provenientes de sua herança, afim de escolher um sistema de ensino que melhor se adequasse à realidade nacional, contratar professores e adquirir laboratórios, para dar início a um acalentado sonho: o

Instituto Elétrico e Mecânico de Itajubá.

E foi, sobre esta aura, realizada a fundação do Instituto, em 1913, onde professores contratados na Bélgica ministravam suas primeiras aulas ainda em francês. A inauguração oficial contou com a presença do Presidente da República e uma discussão acalorada entre o Dr. Theodomiro e o Dr. Paulo de Frontin, sobre os métodos de ensino de engenharia. Sobre isto a frase que pauta os destino da Instituição pode ser resumida pelas palavras de seu Fundador:

"Se a ciência é filha da observação e do experimento é por estes meios que deve ser ensinada."

Posteriormente, em 1956, o Instituto foi federalizado como Entidade de Ensino Superior vinculada ao Ministério da Educação com o nome:

Escola Federal de Engenharia de Itajubá

E a partir de 1972 passou a ser Autarquia de Regime Especial ligada ao MEC.

Cerca de 4.800 engenheiros já se graduaram pela EFEI nos seus 81 anos de existência, e inúmeros são os que se destacam no cenário nacional pelas suas qualidades técnicas, científicas e administrativas. Vários de seus ex-alunos ocupam cargos de destaque, principalmente, nas concessionárias de energia elétrica de nosso país. Pode-se mesmo dizer que a história dos grandes sistemas elétricos brasileiros está intimamente ligada à história da EFEI.

Cerca de 300 estudantes obtiveram o grau de Mestre em Ciências em Engenharia Elétrica e Mecânica pela EFEI, durante os 25 anos de funcionamento de seus cursos de Pós-Graduação, e desempenham suas funções como docentes em diversos

estabelecimentos de ensino ou como profissionais de alta qualificação técnica em diferentes Empresas e Indústrias.

Instalado em uma área de 363.000 m^2, o Campus da EFEI, distante 2 km do centro da cidade de Itajubá, conta com modernas instalações para salas de aula, laboratórios e biblioteca central, além de um moderno Centro Poliesportivo e emissoras de rádio e televisão.

Quanto à organização didático-administrativa, a EFEI é composta de três Institutos de Ensino: Instituto de Ciências, Instituto de Engenharia Elétrica e Instituto de Engenharia Mecânica. Cada Instituto, por sua vez, é constituído por Departamentos de Ensino.

Os Cursos de Graduação são atualmente regidos pela Pró-Diretoria de Graduação e fornecem diplomas nas áreas de Engenharia Elétrica e Engenharia Mecânica.

Os Cursos de Pós-Graduação compreendem disciplinas ministradas pelos Departamentos de Ensino, sendo administrados pelas Coordenações dos Cursos de Pós-Graduação. Estas Coordenações fazem parte da Pró-Diretoria de Pesquisa e Pós-Graduação da EFEI.

Atualmente o Corpo Diretivo da EFEI é formado pelos professores:

Diretor-Geral : Fredmarck Gonçalves Leão
Vice-Diretor: José Carlos Goulart de Siqueira

Diretor do Instituto de Engenharia Elétrica: Felício Barbosa Monteiro
Diretor do Instituto de Engenharia Mecânica: Eduardo Assis Alvarenga
Diretor do Instituto de Ciências: Élcio Rogério Barrak

Pró-Diretor de Graduação: José Policarpo Gonçalves de Abreu
Pró-Diretor de Pós-Graduação: Renato de Aquino Faria Nunes

"Revelemo-nos mais por atos que palavras, dignos de possuir este grande País."

Theodomiro Santiago

Membros do Conselho e Comissão Organizadora

Conselho Nacional de Redes Neurais

Carlos Eduardo Pedreira - PUC-Rio(Presidente)
Dante A.C. Barone - UFRGS
Luiz Pereira Caloba - UFRJ
Renato M. Sabbatini - UNICAMP
Ricardo J. Machado - IBM-Brasil
Teresa Bernarda Ludermir - UFPE

Comissão Organizadora do 1° CBRN

Germano Lambert Torres - EFEI (Presidente)
Edurado Nery - CEMIG
Luiz Pereira Caloba - UFRJ
Luiz Eduardo Borges da Silva - EFEI
Teresa Bernarda Ludermir - UFPE

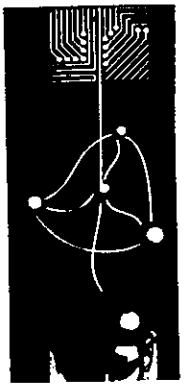
Instrutores da 2° ERN

Teresa Bernarda Ludermir - UFPE (Coordenadora)
Edson Costa de Barros Carvalho Filho - UFPE
Wilson Rosa de Oliveira Jr - UFPE

Comitê Local (Todos da EFEI)

Germano Lambert Torres (Presidente)
Afonso Henriques Moreira Santos
Augusto Nelson Carvalho Viana
Carlos Alberto Dias Coelho
Enio Roberto Ribeiro
Geraldo Lúcio Tiago Filho
Jamil Haddad
José Carlos Grilo Rodrigues
Luiz Augusto Horta Nogueira
Luiz Augusto Ribeiro Salomon
Luiz Eduardo Borges da Silva
Paulo Sizuo Waki

Secretariado: Maria de Fátima Pereira Lima Soares
Regina Maria dos Santos Grilo



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

Índice:

FUNDAMENTOS

NOVAS ARQUITETURAS

ALGORITMOS DE TREINAMENTO

HEURÍSTICAS

AMBIENTES DE DESENVOLVIMENTO

PROCESSAMENTO DE SINAIS I

PROCESSAMENTO DE SINAIS II

PROCESSAMENTO DE IMAGENS

VISÃO COMPUTACIONAL

CONTROLE I

CONTROLE II

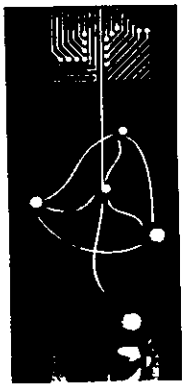
SISTEMAS ELÉTRICOS DE POTÊNCIA

NEUROCIÊNCIA E BIOMÉDICA

MISCELÂNEA I

MISCELÂNEA II

Anotações

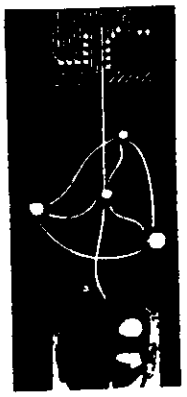


1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

FUNDAMENTOS

Anotações



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajuba
Itajuba. 24 a 27 de outubro de 1994

Neural Particle Discriminator Based on Principal Components Analysis

J.M. Seixas , L.P. Caloba and B. Kastrup
COPPE/EE/UFRJ, C.P. 68504, Rio de Janeiro 21945. Brazil

Abstract

An electron/jets discriminator for high energy physics experiments is developed using neural networks. A principal components analysis is performed in order to reduce dramatically the input space dimension. so that system implementation becomes simpler. It is shown that using simulated calorimeter data. 95% electron efficiency is obtained with 9.5% of jets being misidentified, when input data is projected onto the subspace spanned by six principal components.

Introduction

The LHC (Large Hadron Collider) project [1] is being designed at CERN (Switzerland). It is planned to be operational by the beginning of the next century and it aims to bring new highlights on the study of the fundamental structure of the matter. This is to be achieved by colliding high-energy particles and analyzing the obtained reaction products by means of particle detectors placed all around the collision interaction point. The collision frequency for LHC is expected to be 40 MHz.

Among detectors. calorimeters became very important for collider experiments in the last years [2]. Particles interacting with these detectors deposit entirely their energy and the depositing process is such that calorimeters can also be used to identify the incoming particles. As the signals from the calorimeter can be quite fast. real-time event selection can be achieved. which becomes a desirable feature in the high event rate environment of LHC.

The event selection is performed by complex triggering systems and the operation of such systems is split into successive levels. The first-level trigger (L1) receives all generated events and using mainly the calorimeter information retrieves those events that seem relevant for the physics the experiment is interested on. For LHC. detailed physics simulations show that a second-level trigger system is desired to achieve further event rate reduction after the first-level operation. For the second-level trigger. calorimeters can be combined with other fast detectors. The expected rate at the second-level trigger input is 100 kHz.

The second-level trigger operation is being conceived in two phases [3]. In the first phase, the main features of each detector are extracted. This is performed over the information sent by detectors in a region of interest (ROI) identified by the L1 system. Next, these features are combined by means of a global decision unit and the events are selected according to the processes they represent. A rejection factor of 100 is expected from the whole system.

In this paper we consider the feature extraction problem for calorimeters. The features to be extracted can be found by searching for the best calorimeter variables one could send to the input nodes of a neural network discriminator capable to perform electron/jets separation based only on calorimeter information. For this purpose, Monte Carlo simulations have been used to generate events for the second-level operation [4]. A large sample of QCD jets and single electrons were generated and the interaction of those events with a fine-grained calorimeter was obtained by means of a 20x20 matrix of deposited energy in the ROI. On the sample data, a first-level trigger algorithm [5] was applied. In this manner, a total of 1057 jets and 1634 electrons were produced.

As the outermost cells of the calorimeter sample very few (or null) energy, it was possible to reduce the input space dimension through defining a subregion of 11x11 cells [6]. In order to define this subregion, the cell of maximum energy deposition was found in an event by event basis and the subregion was built around it. Figure 1 shows typical events.

Recent works had explored the neural network approach to perform feature extraction for calorimeters, examining different features [6, 7]. Results revealed that neural discriminators perform better than classical methods. We mention here the matrix and ring methods which were developed on those works and will be referred to in the next sections. The matrix method consists on feeding the input nodes of a neural network with the energy values of the 121 cells that form the subregion of interest. On the other hand, the ring method consists on building concentric rings around the cell of maximum energy deposition, so that this cell becomes the first ring (see Figure 2). The ring sums are obtained by adding up the energy of all cells that belong to a ring. The ring sums are then fed into a network to perform the desired discrimination. For 95% electron efficiency, the matrix and ring methods misclassified 7.2% and 9.7% of the jets, respectively. If one uses a weighting procedure for rings, which has the effect of boosting the energy of the

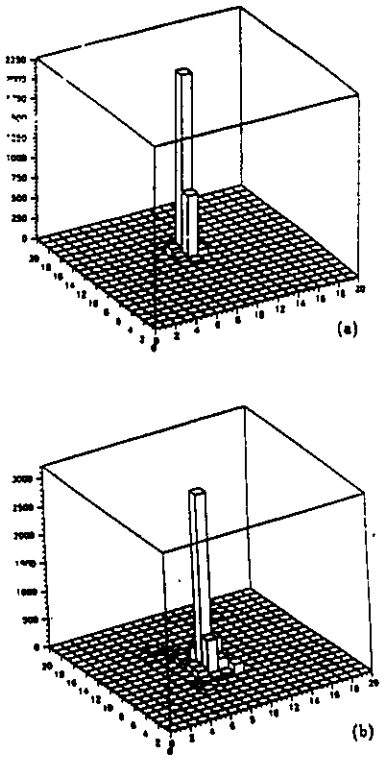


Figure 1. Typical electron (a) and jet (b) events.

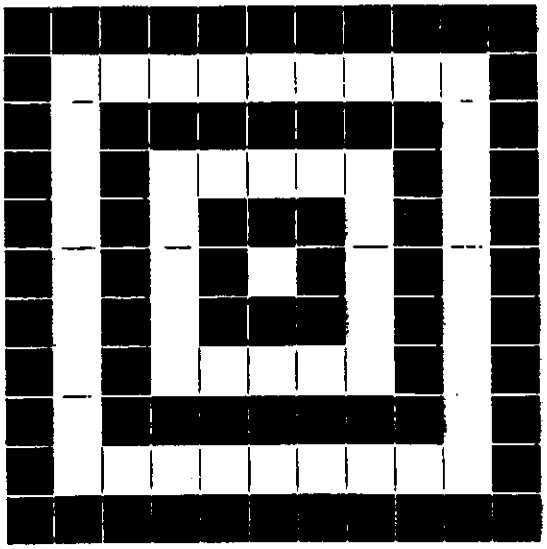


Figure 2. Building rings.

outermost rings, 8.5% of jets are misclassified for the same electron efficiency.

The next section describes the principal component extraction by means of a neural network. Then, it is shown that using at least 4 components, the proposed neural discriminator gets close to the performance obtained with ring sums.

Extracting the Principal Components

The principal components analysis (PCA) appears in many fields of application [8]. The aim is to find a set of M orthogonal vectors in input data space that account for as much as possible of the data's variance. Therefore, the differences in classes of events originally present in the data set can still be identified by projecting the data of the original N-dimensional input space onto the M-dimensional subspace spanned by these vectors. This would perform a dimensionability reduction with the preservation of the information spread around the full input space, as normally M << N.

In the case we are examining, the calorimeter information was reduced to

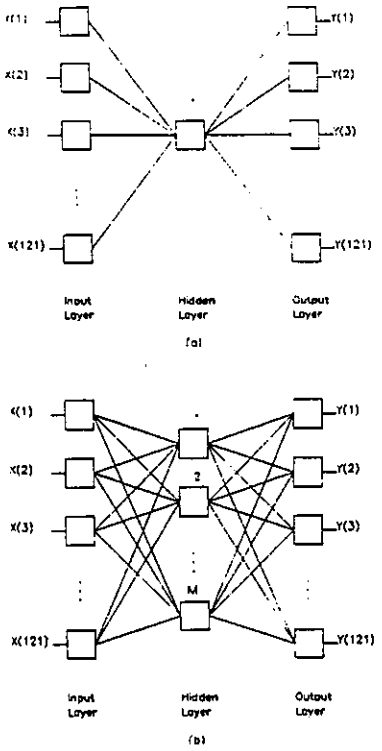


Figure 3. The network topology for extracting the first principal component (a) and the M-th component (b)

a set of 121 energy values that account to the energy deposited on each cell of the detector. Thus, one could think of further reducing this dimensionability by means of PCA, achieving a much more compact discriminator. The ring method mentioned in last section tries to do so using the concept of energy deposition in concentric rings. In fact, using the ring sums as a preprocessing of input data reduces the input space dimension for the discriminator network. As the 11x11 subregion of interest allows building up to six rings (see Figure 2), the input vector for ring analysis has only six components.

The way the principal components are extracted is the following. The first principal component is taken to be along the direction with the maximum variance. The second component is then constrained to lie in the subspace perpen-

dicular to the first and it is taken along the direction of maximum variance. The other components are obtained by following the same procedure.

Neural networks architectures have been used to perform the principal components analysis [9]. Figure 3 shows a set of fully connected networks with one hidden layer used to do so. Each network is linear and has N inputs and N outputs. The first principal component is extracted by using one single unit in the hidden layer (Figure 3a) and training the network so that the output vector is as close as possible to the input vector. The hidden unit ends up revealing the first principal component in its weight vector. The other components are extracted in a similar way, by fixing the weights of the network used to extract the previous component, adding one more unit in the hidden layer and training the remaining network in the same way the first component was obtained (see Figure 3b). At the end of the training procedure, the network had extracted M principal components and the input data can be projected on the M-dimensional subspace spanned by the weight vectors in the hidden layer.

The mean square error (MSE) is the figure of merit in the training phase for deciding how many components one should extract for a given problem. After finding a certain number M of components and observing that MSE does not change significantly by incrementing the number of extracted components, one can consider that the set of M components extracted describes relatively well the distribution of events in the input space.

Results

In order to realize a neural electron/jets discriminator, one neural network was trained with projected events. As the number of principal components are normally much smaller than the dimension of the input space, PCA can be considered as a preprocessing method of input data that enables one to build a more compact discriminator system.

The neural networks used for extracting the principal components and for performing the electron/jets discriminator were simulated with the Jetnet 2.0 package [9]. The training file was built by dividing the total number of events by two. Testing was performed on events that do not belong to the training set. During the training phase of the network that realizes the discriminator, the target value was assumed to be -1 for electrons and 1 for jets. The hyperbolic tangent was the activation function for this network.

After extracting the fourth principal component, the discriminator start to be competitive. Using four components and respective projected data, 95% electron efficiency was achieved with 10.1% of jets misclassified as electrons. For this test, a 4-4-1 neural network performed the discriminator.

An interesting case to be analysed is the use of six principal components. In this case, one can compare the PCA preprocessing with the ring description. Extracting six components, using projected events and a 6-6-1 network topology to realize the discriminator, 9.5% of jets are wrongly classified for the same 95% electron efficiency. This result shows that increasing from four to six components has the consequence of improving performance of the neural discriminator, so that the resulting system reaches the same level of performance obtained by

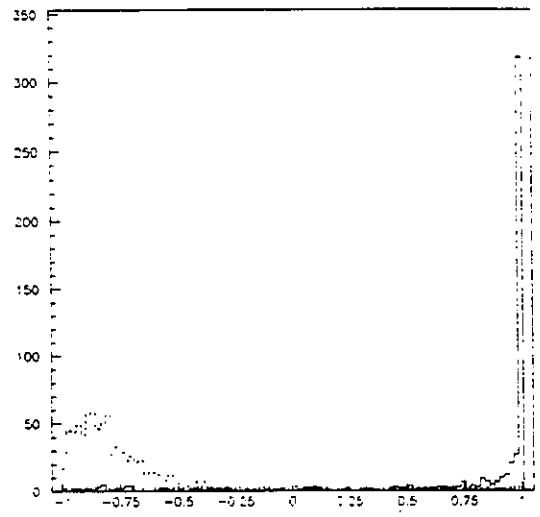


Figure 4. The network output using six principal components.

means of unweighted ring method. One should also keep in mind that the preprocessing needed for PCA is simpler than the one needed for building ring sums. Moreover, digital signal processor technology available nowadays allows the implementation of PCA preprocessing (which means performing inner products) integrated with the neural discriminator. Therefore, in terms of hardware implementation, neural discriminators based on PCA are rather attractive.

Figure 4 shows discriminator's output when six principal components are used. In terms of MSE figure of merit, increasing the number of components from four to six translates into a decrease of the mean square error by a factor of four.

Conclusions

A neural electron/jets discriminator based on principal component analysis was developed. It is based on performing a preprocessing of input data so that each event is projected on the subspace spanned by the principal components. As the number of components required to make the discriminator competitive with other discriminating methods is quite low (four components was the lowest limit found), the proposed discriminator allows a compact design. Moreover, both preprocessing and discrimination can be implemented using a single digital signal processor with the technology available nowadays.

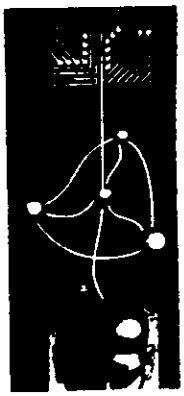
The discriminator being proposed was tested on simulated calorimeter data for the second-level trigger operation on high-energy physics experiments. It was shown that using 6 (4) principal components, 95% electron efficiency is achieved with 9.5 (10.1) % of jets being misclassified as electrons. This performance is similar to the one obtained with a preprocessing that consists on forming ring sums from the original 11x11 matrix of energy deposited in the calorimeter. However, in terms of hardware implementation, the projection of events on the principal vectors is simpler than building ring sums. This makes the neural discriminator based on principal component analysis to be an option for the implementation of a second-level trigger at LHC.

Acknowledgements

We are thankful for the support that has been provided to this work by CNPq (Brazil) and CERN. We would like to thank the RD11 Collaboration for providing the simulated data sets. In particular, we express our best thanks to R.K. Bock, F. Block, W. Krischer, I.C. Legrand and J. Carter, all from CERN, for fruitful discussions concerning this work, and A. Ordacgy for helping in part of the analysis.

REFERENCES

1. The Atlas Collaboration. Letter of Intend for a General-Purpose pp Experiment at the Large Hadron Collider at CERN. CERN/LHCC/92-4, 1992.
2. R. Wigmans - Advances in Hadron Calorimetry. Rev. Nucl. Part. Sci. 41 (1991) 133
3. J. Badier et al. IEEE Trans. on Nuclear Science **40** (1993) 45.
4. G. Klyuchnikov et al. A Second-level Trigger Based on Calorimetry Only. **CERN-ATLAS/DAQ-007** (1992).
5. RD-27 Collaboration. **CERN-ATLAS/DAQ-005** (1992).
6. J.M. Seixas et al. A Second-Level Trigger System Based on Calorimeters and Using Neural Networks for Feature Extraction and Electron/Jets Discrimination. IV Int. Conf. on Calorimeters for High En. Phys. Elba, Italy, 1993.
7. J.M. Seixas et al. Neural Networks Applied to a Second-Level Trigger System. III Int. Workshop in Softw. Eng., Artificial Intell. and Expert Sys. for High-En. and Nucl. Phys. Oberammergau, Germany, 1993.
8. S. Haykin. Adaptive Filter Theory. Prentice-Hall (1991)
9. J. Hertz, A. Krogh and R.. G. Palmer. Introduction to the Theory of Neural Computation. Addison-Wesley (1991).
10. L. Lönnblad et al. Comput. Phys. Commun. **70** (1992) 167.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajuba
Itajuba, 24 a 27 de outubro de 1994

Issues on the Complexity of Training Weightless Neural Networks

Marcílio C. P. de Souto
Katia S. Guimarães
Teresa B. Ludermir

Universidade Federal de Pernambuco
Departamento de Informática
Cx. Postal 7851 - 50.732-970 - Recife - PE - Brazil
{mcps,katia,tbl}@di.ufpe.br

Abstract. In this paper, it is extended the Judd's results with respect to learning computational complexity of weighted neural models to include the weightless neural models. It is shown, for example, that also is NP-complete any algorithm that aims to *load any* performable training set in *any* conceivable weightless neural network. It is also conjectured that a specific architecture class, *pyramidal* architectures (for the weightless models), may be a way to overcome the NP-completeness of learning.

1 Introduction

One of the most important features of neural networks[1] is their ability to generalize to new situations. Once trained, a network will compute an input/output mapping which, if the training data was representative enough, will closely match the unknown rule which produced the original data.

The work in this paper deals with basic theoretical questions regarding learning by neural networks, it was inspired by Judd ([7]) who shows the following problem to be NP-complete:

> "Given a neural network and a set of training examples, does there exist a set of edge weights for the network so that the network produces the correct output for all training examples?"

this problem is called the **loading problem**.

Judd developed his work based on McCulloch-Pitts (MPC) neurons ([10]). MCP neurons are implemented by threshold logic gates, where variable input weights play a role analogous to that of synapses in natural neurons. The models used in this paper is based on a different model called the weightless neuron model ([1]). The weightless model is based on the simple operations of look-up table which is best implemented by random access memory (RAM) and where knowledge is directly "stored" in the memory (the look-up tables) of the nodes during learning. Some advantages of this model are: (1) it is straightforward to implement in hardware; (2) learning is not unreasonably slow and (3) error-correction requires only global success signal ([9]).

It is important to point out that there is no study on learning complexity of weightless models, being this paper the first effort towards putting these models in the context of learning computational complexity. Another question is that this kind of study can help to design neural networks, because it identifies underlying problems in learning and tries to find ways to avoid them. Thus, these can yield techniques to neural network design.

Some background on computational complexity is necessary for a better understanding of this work. Here, whenever it is said polynomial time it is meant polynomial time in the length of any binary encoding of the input and problems approached here are always decision problems ([5]).

A problem is in class P when there is a polynomial time algorithm which solves the problem. A problem is in NP when a "guessed" solution for the problem can be verified in polynomial time. A problem (set) H is NP-hard iff for each problem (set) Q in NP, there is a polynomial time transformation f_Q from Q to H, such that given any instance I of Q, $I \in Q$ iff $f_Q(I) \in H$. Then, a problem is NP-complete iff it is both NP and NP-hard. Examples of NP-complete problems are: the Boolean satisfiability problem, the traveling salesperson problem, the set-splitting problem.

The remainder of this paper is divided into three sections. Section 2 presents the weightless neural models and their main characteristics. The main section of this work is Section 3, which puts the weightless models in the context of computational complexity and Judd's work is used as base to discussion ([7]). It is also studied issues regarding loading *pyramidal* archi-

[1] In this paper, the term "neural network" always means feed-forward ones which have binary inputs/outputs and the learning paradigm analyzed is the supervised learning.

tectures, and it is made a parallel between theoretical and empirical results in complexity of learning. To develop this parallel are the backpropagation (weighted models) and pyramidal (weightless models) architectures used. Finally, the last section summarize the discussion in support to our conjecture.

2 Weightless Neural Models

Definition 1 A RAM Neural Network is an arrangement of a finite number of neurons in any number of layers, in which the neurons are RAM (Random Access Memory) nodes. A RAM node is represented in the Figure 1.

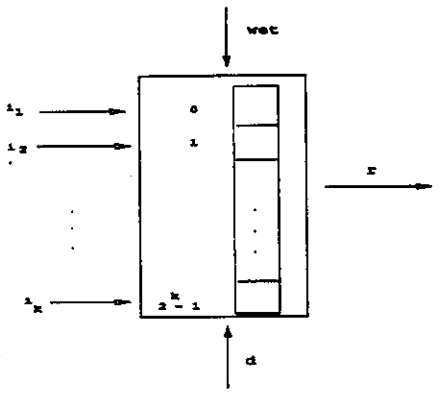


Figure 1: RAM node

The input to each neuron may be external input or outputs of neurons from another layer. The data out may be 0 or 1. The set of connections is fixed and there are no weights in such nets. Instead, the function performed by the neuron is determined by the contents of the RAM - its output is the value contained at the activated memory location. There are 2^{2^N} different functions which can be performed on N address lines and these correspond exactly to the 2^N states that the RAM can be in, that is, a single RAM can compute *any Boolean function* of its inputs.

Seeing a RAM node as truth table (look-up table) the output of the RAM node is described by Equation (1) below:

$$r = \begin{cases} 0 & if\ C[p] = 0 \\ 1 & if\ C[p] = 1 \end{cases} \qquad (1)$$

where C[p] is the contents of the address position associated with the input pattern p.

Definition 2 A PLN Neural Network is an arrangement of a finite number of neurons in any number of layers, in which the neurons are PLN (Probabilistic Logic Node) nodes.

A PLN node differs from a RAM node in the sense that a 2-bit number (rather than a single bit) is now stored at the addressed memory location. The contents of this location (C[p]) can be 0, 1, or u , and it represents one of three possibilities (0, 1, 0.5, respectively) of firing (i.e. generating a 1) at the output. The output of the PLN nodes described by Equation (2) below:

$$r = \begin{cases} 0 & if\ C[p] = 0 \\ 1 & if\ C[p] = 1 \\ random(0,1) & if\ C[p] = u \end{cases} \qquad (2)$$

where C[p] is the contents of address position associated with the input pattern p and random(0,1) is a random function that generates zeros and ones with the same probability.

Besides the RAM and PLN nodes there are many variations and extensions of the RAM node (e.g., MPLN ([11]), cut-point ([9]), GSN ([4]), called RAM-based nodes.

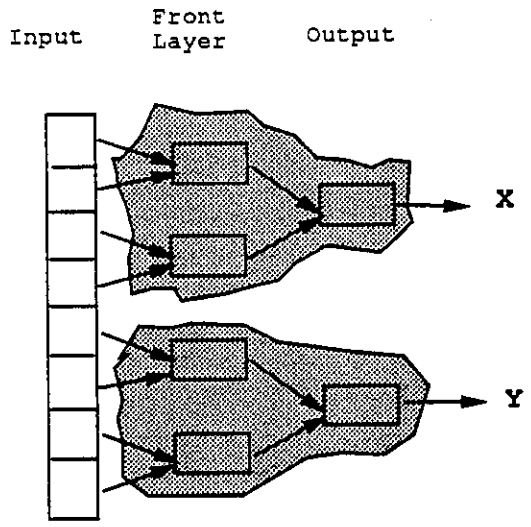


Figure 2: Example of a pyramidal architecture. The shaded and outlined area surrounding output nodes X and Y encompass all the nodes in each support cone. Note that each support cone correspond to an individual pyramid in the architecture.

Although there are several kinds of RAM-based nodes, the architecture used in most experiments developed is the *pyramidal* one (Figure 2). In this topology, neurons are arranged in hierarchical non-overlapping pyramids (trees), where each pyramid culminates in a

single output line. Therefore, if more than one output is associated with the problem, there will be one pyramid for each. As it has already mentioned, the number in the storage locations of RAM-based networks increases exponentially (2^N per node, where N is the size of the node input) with the size of the input problem, then the use of pyramidal architecture allows a decrease in this number.

3 Weightless Models and Complexity of Learning

Judd's work ([7]) was pioneer in the field of learning computational complexity of artificial neural networks. Before his work there was no characterization of learnability in terms of its computational complexity. Notwithstanding, the weightless models still lack this kind of characterization.

3.1 Judd's work in the context of Weightless Models

Judd studies several classes of architectures and shows each one of them to be NP-complete with respect to the loading problem. He does that by producing for each architecture, in its respective class, training examples such that any algorithm performs poorly on some networks and training set in that class (any instance of 3-Satisfiability problem can be transformed into polynomial time to some architecture and training set in this class). The more general case studied precludes only the most ambitious interpretation of the goal in connectionist learning. That is, the connectionist belief of finding an algorithm that is guaranteed to load *any* performable task in *any* conceivable network.

The results achieved are negative, because in that more general case (as in almost all subcases analyzed) the loading problem stays NP-complete. The only tractable case studied is, however, trivial, for it is a class of architectures (with *support cone interaction graph* having limited *armwidth* [7, 12]) which seems useless to practical activities. Nevertheless, the NP-completeness results define only the upper bound of the loading problem. Thus, they do not make impossible that in the average case it may be resolved in polynomial time.

It is interesting to verify that all Judd's results found in ([7]), which are related to learning computational complexity of weighted neural models, are also extendable to weightless neural models. This is because, the proofs found there are independent of any particular training algorithm and they are based on the set of Boolean functions[2]. A crucial point in those proofs, which allows this extension, is that neurons are considerate like truth tables (look-up tables). Thus, those proofs are directly extensible to weightless neural model, since weightless neuron models are truth tables as well. Examples of theorems and corollaries extended are:

> **Theorem 3** Loading weightless neural networks, whose node function set are only AND and OR functions, is NP-complete.
>
> **Corollary 3** Loading is NP-complete, independent if one are using weighted or weightless neural models.

This Judd's way to approach the loading problem was criticized by Blum *et. all* ([2]) and Dasgupta *et. all* ([3]) who proposed a study of the loading problem in terms of a *specific* neural network and node function set. Nevertheless, most of their results lead to the NP-completeness of learning.

3.2 Loading pyramidal architectures

One fundamental point not approached until now is the issue of loading deep networks. This issue usually are not analyzed in the weighted models, because the connectionist literature uniformly reports great hardness in loading these kinds of networks ([13, 8]).

In the context of weightless neural models, though, it is interesting to consider deep networks, since the architecture mostly used is the pyramidal, which tends to have high depth.

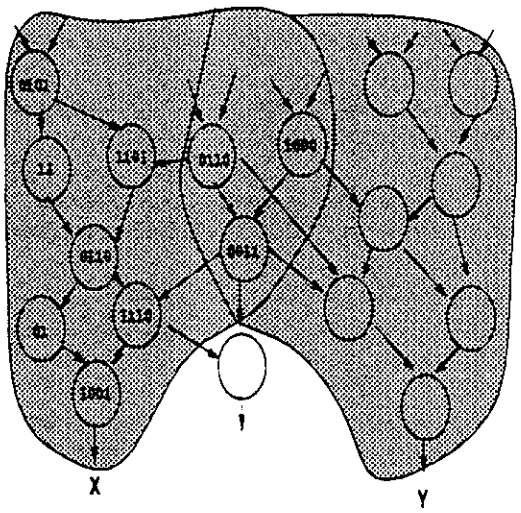


Figure 3: Illustration of support cones. The shaded and outlined area surrounding output node X encompasses all the nodes in its support cone. Likewise the support cone for output Y is shaded in.

[2] And-Or functions, linearly separable functions, all Boolean functions, etc.

Pyramidal networks form a special class of architectures. Given a network of this class it is important to verify that its support cones[3] do not overlap among them. This is a very important characteristic, once one of the causes to the hardness of learning is the interaction among support cones. Such interaction allows that constraints of choosing a configuration of functions in a determined support cone interferes in the choice of the configuration in the other overlapping support cone.

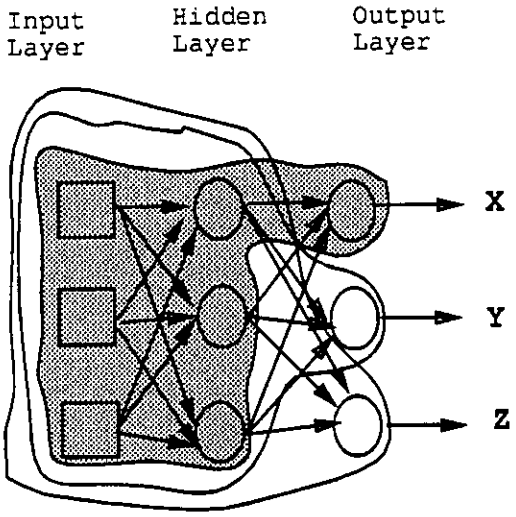


Figure 4: Example of a backpropagation architecture. The shaded and outlined area surrounding output node X encompasses all the nodes in its support cone. All the support cones in this architecture have the same nodes with exception of the output nodes, which is different to each support cone.

These negative results about the complicated interaction among support cones can be verified empirically, for example, looking at the features of the well-known and widely used backpropagation architectures ([6]). These architectures usually are fully connected, that is, a determined neuron of a level is connected to all neurons of the former. Therefore, they have the worst case in the interaction among support cones (Figure 4), for each support cone overlaps to all others. Empirically, it is acknowledged that as the network gets larger and deeper, the amount of time required for them to load the training data grows prohibitively ([8]). Looking at these results, in terms of backpropagation architectures, they reflect the increasing in the size of support cones. Such increase produces an explosion in the number of constraints that is necessary to deal with, hence in these networks learning becomes very hard.

On the other hand, there are experiments which weightless neural model using pyramidal architectures, when compared to backpropagation methods, can learn orders of magnitude faster ([11]). The weightless neural models often use pyramidal architectures, where each pyramid sees a part of the total input and there are no overlappings among pyramids (Figure 2). Consequently, each output neuron has its own support cone (in the case the total pyramid which it belongs to) and the choice of its support cone configuration will not interfere with the other support cones. Thus, in these architectures there are less constraints to deal with (since that the only concern is about individual pyramids) than the myriad of them that exist in the backpropagation architectures, such that learning becomes easier.

4 Final remarks

In this paper it has been sketched the first connections between the theory of computational complexity and weightless neural models learnability. The learning computational complexity in the weighted models have been reviewed, and one can verify that, in most cases, the results lead to the NP-completeness. With base on Judd's work these results were extended to include the weightless models.

Also, a parallel was made between backpropagation (weighted neural models) and pyramidal (weightless neural models) architectures in terms of interactions among support cones. It has been verified that there are experiments which show that weightless neural models using pyramidal architecture can learn orders of magnitude faster than the backpropagation ones.

Based on this it is conjectured that the class of pyramidal architectures may be a way to overcome the hardness of learning and it is being investigated if this is true. It is important to point out that the characterization of a polynomial architecture class (an architecture class that all tasks it can perform are loading in polynomial time) is a fundamental guide to neural networks design.

[3] This is the set of all nodes that can affect the behavior of an output node (Figure 3) [7]. Note that in Figure 3 the two cones overlap in three nodes. The binary numbers name node functions and as group they constitute a partial configuration for output node X.

References

[1] I. Aleksander and H. Morton. *An introduction to neural computing.* Chapman and Hall. London. Great Britain, 1 edition, 1990.

[2] A. L. Blum and R. L. Rivest. Training a 3-node neural network is NP-complete. *Neural Networks*, 5:117–127, 1992.

[3] B. Dasgupta, Siegelmann. H. T., and E. Sontag. On the complexity of training neural networks with continuos activation functions. Technical report, Rutgers University, New Brunswick, NJ, December 1993.

[4] C. B. C. E. Filho. *Investigation of Boolean Neural Network Based on a Novel Goal-Seeking Neuron.* PhD thesis, University of Kent at Canterbury, 1990.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, San Francisco, California, 1979.

[6] R. Hecht-Nielsen. Theory of the backpropagation neural network. In *Anais of International Joint Conference on Neural Network*, Washington, June 1989.

[7] J. S. Judd. *Neural network design and the complexity of learning.* The MIT Press, Cambridge, Massachusetts, USA, 1990.

[8] J. F. Kolen and A. K. Goel. Learning in parallel distributed processing networks: computational complexity and information content. 1990. To appear in IEEE Systems, Man, and Cybernetics.

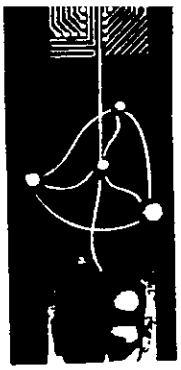
[9] T. B. Ludermir. Learning algorithms for cut-point neural networks. In *Anais do X SBIA*, pages 433–443, Porto Alegre, RS, October 1993.

[10] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. In J. A. Anderson and E. Rosenfeld, editors, *Neurocomputing: Foundations of Research*, chapter 2, pages 18–27. The MIT Press, USA, 5 edition, 1990.

[11] C. Myers and I. Aleksander. Learning algorithms for probabilistc neural nets. In *IEE International Conference on Artificial Neural Networks*, pages 310–314, London, UK, October 1989. IEE.

[12] N. Robertson and P. D. Seymour. Graph Minor. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[13] G. Tesauro and R. Janssens. Scaling relationships in backpropagation learning: dependece on predicate order. *Complex Systems*, 2:39–44, 1988.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajuba
Itajuba, 24 a 27 de outubro de 1994

Especificando Formalmente Redes Neurais via MOFEU

Débora Abdalla
Edson Costa de Barros Carvalho Filho

Universidade Federal de Pernambuco
Departamento de Informática
Cx. Postal 7851 - 50.732-970 - Recife - PE - Brazil
{das,ecdbcf}@di.ufpe.br

Sumário. Este artigo apresenta uma visão geral do modelo formal de especificação universal para redes neurais, MOFEU. O modelo é representado por uma estrutura matemática que permite uma especificação a nível de neurônio, arquitetura, aprendizagem e modo de operação. Um aspecto temporal é fornecido no sentido que a configuração e o estado são dinamicos e podem ser instanciados em um dado instante de tempo. Os modelos de redes neurais que poderem ser especificados via MOFEU são chamados de redes neurais bem formadas. Argumenta-se que este modelo seja geral o suficiente para representar uma grande variedade de paradigmas. É mostrada, ainda, a equivalencia das redes neurais bem formadas com sistemas normais dinamicos. Desta forma, algumas propriedades encontradas na teoria geral de sistemas são trazidas para os sistemas neurais.

1 Introdução

Neste artigo é apresentado um modelo de especificação formal para redes neurais. A motivação para tal pesquisa se deve ao fato do grande número de trabalhos em redes neurais que vem sendo progressivamente desenvolvidos [10, 11, 7, 6, 1, 5] e na constatação de que a maioria dos seus resultados são baseados em simulações computacionais. A falta de formalização da especificação das ações dos vários paradigmas de redes neurais contribui para o mal entendimento do funcionamento e comportamento das redes. O processo de análise das propriedades inteligentes (comportamento) das redes é o primeiro a sofrer com esta carencia de formalismo, uma vez que para verificar se um dado modelo de rede possui um dado comportamento faz-se necessário a implementação e simulação da rede.

As técnicas de especificação formal estão tendo um grande impulso no desenvolvimento de software, as técnicas para especificação e desenvolvimento de aplicações em Inteligencia Artificial não estão bem definidas até o presente estado da arte [9]. Uma formalização e uma sólida fundamentação são essenciais para auxiliar a área de redes neurais alcançar sua maturidade [4]. Um modelo matemático para uma teoria unificada de aprendizagem neural para várias arquiteturas de redes neurais pode ser encontrado em [8]. O modelo é aplicável a um único neurônio, a uma rede de camadas e a um sistema neural auto-organizável com conexões retroativas. Assim, tenta-se estabelecer alguns fundamentos matemáticos na neurocomputação.

O MOdelo Formal de Especificação Universal para redes neurais - MOFEU, procura definir uma estrutura matemática que permita especificar de forma universal a maior quantidade possível de paradigmas de redes neurais. Tal modelo, além de se prestar para especificação universal, apresenta a caracteristica altamente importante de que os paradigmas de redes neurais especificados neste modelo são equivalentes a um sistema normal dinamico.

O uso de especificações formais, em redes neurais, irá facilitar a automatização do processo de construção, animação e análise do comportamento de redes neurais. Uma segunda abordagem, para fazer o uso da especificação formal, é a inferencia de propriedades fundamentais da rede utilizando provas formais sem a necessidade de simulação, buscando assim, resultados mais consistentes e confiaveis do que a simulação.

2 A Proposta do Modelo Formal

Pode-se datar entre os anos próximos de 1950 uma mudança no pensar cientifico [2]. Começava a se ressaltar o valor das teorias, em particular as formuladas com o auxílio da matemática. Com o uso da linguagem matemática, pretendia-se combater a falta de precisão e clareza nas formulações científicas. Difundiu-se, então, a construção de objetos-modelos e modelos teóricos. Um modelo não é somente uma descrição abstrata da realidade, mas também uma simplificação que desconsidera algumas caracteristicas essenciais. Primeiro observa-se o objeto de estudo e realiza-se as primeiras conjecturas. Diante das informações recolhidas deve-se identificar as que são relevantes e ordená-las. Como resultado obtem-se objetos-modelos ou modelos conceituais que poderão dar uma imagem simbólica do

real.

A vantagem de se ter modelos conceituais é a contribuição dada ao entendimento e explicação da realidade. Os modelos formais exigem uma análise rigorosa e exata, portanto, eles conduzem a uma abordagem mais crítica e uma melhor definição da realidade. Além disto, na medida que teorias e experiências são contestadas ou enriquecidas, elas são desfeitas e substituídas por outras mais convenientes.

O modelo formal proposto, MOFEU, objetiva representar o maior número possível de diferentes paradigmas. Para encontrar tal modelo genérico é necessário o conhecimento e análise dos vários paradigmas existentes e um posterior processo de abstração onde características comuns serão selecionadas. Como resultado, é obtido um conjunto de conceitos fundamentais os quais formarão a base para a construção de um modelo onde qualquer paradigma de rede neural será considerado uma de suas instâncias. Um modelo com tal propriedade será chamado *universal*. Assim, o modelo define os elementos essencias para que uma rede neural apresente comportamento inteligente. Estes elementos serão chamados de *canônicos* e estarão representados através de ferramentas matemáticas, tais como: conjuntos, relações e funções. Desta forma espera-se apresentar um modelo claro e preciso, definindo-se uma metodologia formal para descrever neurônios, arquiteturas e processos de aprendizagem.

Tem-se, então, que uma rede neural $\mathcal{NN}$ é definida através da sequinte estrutura:

$$\mathcal{NN} = \langle N, T, D, F, D_0 \rangle$$

onde:

N — Conjunto contendo todos os neurônios que compõem a rede. Cada neurônio é identificado por um índice de forma que um neurônio i é referenciado por n_i. Assim, o número total de neurônios é dado pelo índice n. $\mathrm{N} = \{n_1, n_2, ..., n_n\}$.

T — Conjunto de pares ordenados de neurônios, ou seja, uma relação sobre N determinando a topologia da rede. Qualquer conexão entre dois neurônios deve estar representada por um par ordenado em T. Por exemplo, se a saída de n_i é entrada para n_j então o par (n_i, n_j) pertence a T. $T \subseteq \{(n_i, n_j) \in (N \times N)\}$.

D — Domínio da rede. Conjunto formado por 3 conjuntos, W, A e Θ. Define uma faixa de valores sobre a qual o estado e a configuração podem variar a partir do tempo inicial ($t = 0$) até um tempo final ($t = m$). O estado da rede é definido como a saída de todos os neurônios em um dado instante. A configuração é definida como o estado da rede, mais os valores de todos os pesos e memórias locais. $D = \{W, A, \Theta\}$.

O conjunto W representa todos os pesos associados às conexões pertencentes a T para qualquer instante do intervalo de tempo definido. $W = \{w_{ij}(t) \mid w_{ij}(t) \in \mathbb{R} : t = 0, 1, ..., m \text{ e } (n_i, n_j) \in T\}$.

O conjunto A representa todos os valores de saída (ativação) dos neurônios pertencentes a N. Contém todos os estados pelo qual a rede pode passar durante o intervalo de tempo definido. $A = \{a_j(t) \mid a_j(t) \in \mathbb{R}$; $t = 0, 1, ..., m$ e a_j é a saída de $n_j\}$.

O conjunto Θ representa todos os valores da memória local dos neurônios pertencentes a N para qualquer instante do intervalo de tempo definido. $\Theta = \{\theta_j(t) \mid \theta_j(t) \in \mathbb{R}; \mathrm{t} = 0, 1, ..., \mathrm{m} \text{ e } \theta_j$ é a memória local de $n_j\}$.

F — Conjunto de funções que mudam o estado e a configuração da rede. Define, também, o modo de operação. $F = \{F_A, F_W, F_\Theta, F_O\}$.

A função F_A é a função de ativação do neurônio. Nela está definido como um neurônio produz uma saída. A função tem como domínio o conjunto de neurônios N e o conjunto $\mathbb{N}$ dos números naturais representando o tempo. A imagem é o conjunto dos valores de saída A. Se a função é aplicada a um neurônio no instante de tempo t, uma saída será produzida no instante $t + 1$. Assim, F_A é definida pela seguinte assinatura: $F_A : N \times \mathbb{N} \rightarrow$ A tal que $F_A(n_j, t) = a_j(t + 1)$.

A função F_W é a função pela qual os pesos são alterados. Seu domínio é o conjunto T que contém todas as conexões da rede e o conjunto $\mathbb{N}$ dos números naturais representando o tempo. A imagem é o conjunto W dos pesos associados a cada conexão. Se a função é aplicada a uma conexão no instante de tempo t o seu peso terá um novo valor no próximo instante $t + 1$. Então, a assinatura de F_W é definida por: $F_W : T \times \mathbb{N} \rightarrow$ W tal que $F_W((n_i, n_j), t) = w_{ij}(t + 1)$.

A função F_Θ é a função que aplicada a um neurônio muda o valor da memória local. O domínio é o conjunto de neurônios N e o conjunto $\mathbb{N}$ dos números naturais representando o tempo. A imagem é o conjunto Θ contendo o valor da memória local de cada neurônio em qualquer instante do intervalo de tempo de operação da rede. A assinatura de F_Θ é definida da seguinte forma: $F_\Theta : N \times \mathbb{N} \rightarrow \Theta$ tal que $F_\Theta(n_j, t) = \theta_j(t + 1)$.

As funções F_A, F_W e F_Θ dizem como o estado e a configuração da rede mudam de valor. Elas são definidas para atuar em um único neurônio ou uma única conexão. A simulação do funcionamento da rede é dada por um conjuto de 3 funções, onde cada uma delas diz como as tres funções anteriores devem ser aplicadas. Então, F_O é um conjunto de 3 funções, F_{OA}, F_{OW} e $F_{O\Theta}$, definindo o modo de operação da rede. $F_O = \{F_{OA}, F_{OW}, F_{O\Theta}\}$.

A função F_{OA} informa que um grupo de neurônios

deve produzir uma saída. Isto é realizado aplicando-se a função F_A a cada neurônio de um subconjunto de N. Seja 2^N o conjunto das partes de N. Rigorosamente, F_{OA} é definida pela seguinte assinatura: $F_{OA} : 2^N \times \{F_A\} \rightarrow 2^A$. Como o conjunto de funções é unitário, para efeitos de simplificação, será utilizado apenas F_A em substituição à $\{F_A\}$. Esta observação será mantida por todo o texto. Assim, tem-se: $F_{OA} : 2^N \times F_A \rightarrow 2^A$.

A função F_{OW} modifica os pesos das conexões de um subconjunto de T. Indica que a função F_W será aplicada a cada par (n_i, n_j) do subconjunto. Seja 2^T o conjunto das partes de T. A assinatura de F_{OW} é dada por: $F_{OW} : 2^T \times F_W \rightarrow 2^W$.

A função $F_{O\Theta}$ define um grupo de neurônios que terão a memória local modificada. Determina que a cada neurônio de um subconjunto de N será aplicado a função F_Θ. Seja 2^N o conjunto das partes de N. Então, a assinatura de $F_{O\Theta}$ é definida por: $F_{O\Theta} : 2^N \times F_\Theta \rightarrow 2^\Theta$.

D_0 — Domínio inicial da rede. Conjunto formado por 3 conjuntos, $A(0)$, $W(0)$ e $\Theta(0)$. Define os valores para o estado inicial e a configuração inicial da rede. $D_0 = \{W(0), A(0), \Theta(0)\}$.

O conjunto $W(0)$ contém todos os pesos no instante $t = 0$. $W(0) = \{w_{ij}(0) \mid w_{ij}(0) \in W\}$.

O conjunto $A(0)$ contém a saída de todos os neurônios no instante $t = 0$. $A(0) = \{a_j(0) \mid a_j(0) \in A\}$.

O conjunto $\Theta(0)$ contém os valores da memória local de todos os neurônios no instante $t = 0$. $\Theta(0) = \{\theta_j(0) \mid \theta_j(0) \in \Theta\}$.

Em resumo, o modelo proposto apresenta-se na forma de tupla, onde a representação universal é obtida via elementos canônicos que são utilizados para especificar neurônio, arquitetura, aprendizagem e modo de operação [12]. Ressalta-se que tal estrutura é representada em termos matemáticos pela utilização de conjuntos, relações e funções. Existe também um cuidado especial na notação. Tentou-se, na medida do possível, utilizar símbolos e expressões que apresentassem uma relação direta com seu significado.

A estrutura matemática proposta para representar modelos de redes neurais, tem se prestado para vários modelos, alguns exemplos de paradigmas especificados via MOFEU podem ser encontrados em [13]. É possível que algum modelo abstrato de rede neural não possa ser especificado pela estrutura universal proposta, neste caso, seria necessário um poder de especificação maior, que pode ser obtido incluíndo novos elementos canônicos ao modelo. Os paradigmas de redes neurais que podem ser especificados pelo modelo proposto, serão chamados de *redes neurais bem formadas.*

3 Equivalência de Redes Neurais Bem Formadas e Sistemas Normais Dinâmicos

A abordagem sistemica clama que qualquer fenômeno, seja ele natural ou artificial, é em sua essência um sistema. E como tal, seu entendimento é realizado pela análise das relações e integração entre os componentes. Uma tendência em teoria geral de sistemas é desenvolver métodos que facilitem a construção de sistemas conceituais onde as interações entre os elementos, não necessariamente todas, sejam suficientemente incorporadas.

Uma rede neural é em sua essência um sistema, onde as propriedades inteligentes emergem da composição de unidades menores e mais simples, os neurônios. Assim, foi adotada a teoria geral de sistemas como um caminho favorável a uma especificação formal. A ênfase dada por esta teoria às relações para o entendimento e análise do comportamento de sistemas complexos é o ponto principal para estabelecer uma associação com redes neurais e partir para a compreensão dos diversos paradigmas existentes.

O objetivo é demonstrar a equivalência entre uma *rede neural bem formada* e um *sistema normal dinâmico*. A partir da definição matemática de um sistema geral [3] serão colocadas propriedades adicionais até chegar à definição de sistemas normais dinâmicos. A seguir será apresentada a definição de um sistema geral, um sistema normal e um sistema dinamico, segundo [3], com as devidas demosntrações para redes neurais bem formadas. Uma apresentação detalhada da demonstração da equivalência entre redes neurais bem formadas e sistemas normais dinâmicos pode ser encontrada em [14].

Definição 1 *Um sistema geral é um par ordenado (E, R), onde E é um conjunto de objetos do sistema e R um conjunto de todas relações possíveis entre os objetos de E.*

$$\emptyset \in R \subseteq P(\cup_{n \in Ord} E^n)$$

onde Ord é a clase de todos os números ordinais, E^n o produto cartesiano de E e $P(\cup_{n \in Ord} E^n)$ a coleção de de todos os subconjuntos de $\cup_{n \in Ord} E^n$.

Proposição 1 Dada a definição de um sistema geral (Definição 1), uma rede neural bem formada, $\mathcal{NN}$, é um par ordenado (E', R'). Onde, E' é o conjunto dos objetos do sistema e R' é o conjunto de relações entre eles. Os objetos do sistema são os neurônios e as variáveis. Os neurônios serão representados por um conjunto N. As variáveis serão representadas por três classes de variáveis: um conjunto W de variáveis contendo o peso de cada conexão entre os neurônios.

um conjunto A de variáveis contendo o valor de saída de cada neurônio e um conjunto Θ de variáveis contendo o valor da memória local de cada neurônio. O conjunto de relações R' é constituído de dois tipos de relações: uma relação C estabelecendo as conexões entre os neurônios e um conjunto F de relações sobre as variáveis. Para cada classe de variáveis é definida uma relação sobre ela. Estas relações serão denotadas por F_W, F_A e F_Θ. Assim, tem-se que uma rede neural, $\mathcal{NN} = (E', R')$, é um sistema geral, onde:

(i)

$$\begin{aligned} E' &= \{N \cup W \cup A \cup \Theta\} \\ N &= \{n_j | n_j \text{ é um neurônio }, j \in \mathbb{N}\} \\ W &= \{w_{i,j} | w_{i,j} \text{ é uma variável que associa pesos a uma conexão entre } n_i \text{ e } n_j\} \\ A &= \{a_j | a_j \text{ é uma variável que contém os valores de saída de um neurônio } n_j\} \\ \Theta &= \{\theta_j | \theta_j \text{ é uma variável com os valores da memória local de um neurônio } n_j\} \end{aligned}$$

(ii)

$$\begin{aligned} R' &= \{ C , F \} \\ C &\subseteq N \times N \\ F &= \{F_W \cup F_A \cup F_\Theta\} \end{aligned}$$

Definição 7 *Um sistema geral $S = (E, R)$ é* um sistema normal *se somente se:*

(a) $R = \{C, F\}$, *onde C é uma relação estruturada e F é uma relação comportamental.*

(b) *C é uma relação de conexão, isto é: $C \subset E \times E$.*

(c) *F é uma relação comportamental global, isto é:* $F \subset \times\{Rx_i : (\forall x_i \in E) \rightarrow (x_i = \textit{variável})\}$

Proposição 4 Uma rede neural bem formada $\mathcal{NN} = (E', R')$ é um sistema normal.
Demonstração: Para mostrar que uma rede neural é um sistema normal, basta mostrar que ela satisfaz os três itens requeridos pela Definição 7, então tem-se:

(a) $R' = \{C, F\}$, onde C é uma relação estruturada e F é um conjunto de relações comportamentais (estas afirmações foram demosntradas como verdadeiras pela Proposição 2 encontrada em [14]).

(b) $C \subset N \times N$ é uma relação de conexão (esta afirmação é demosntrada como verdadeira pela Proposição 3 encontrada em [14]).

(c) A relação $F = \{F_W, F_A, F_\Theta\}$ é um conjunto de relações comportamentais sobre o conjunto de variáveis W, A, Θ (Proposição 2, ver [14]). Sejam $Rw_{i,j}$, Ra_j, $R\theta_j$ o domínio das variáveis $w_{i,j}$, a_j, θ_j respectivamente e cada F_i definida da seguinte forma: $F_W \subset \times\{Rw_{i,j} : w_{i,j} \in W\}$, $F_A \subset \times\{Ra_j : a_j \in A\}$ e $F_\theta \subset \times\{R\theta_j : \theta_j \in \Theta\}$. Pela Definição 6, (ver [14]), cada função F_i é uma relação comportamental global.

Dito isto, uma rede neural bem formada é um sistema normal como se pretendia demonstrar.

Definição 10 *Seja $S = (E, R)$ um sistema geral. Seja Ω um conjunto de todas as trajetórias sobre $\mathcal{D}$. A 4-dupla $\mathcal{DS} = (E, R, \Omega, \rho)$ é um* sistema dinâmico *se somente se:*

(a) *E é um conjunto de variáveis.*

(b) *R é uma relação comportamental global.*

(c) *ρ é uma relação entre Ω e $P(R)$, $\rho \subset \Omega \times \mathcal{P}(R)$, tal que para cada par $[(T, \leq), B] \in \rho$, existe um mapeamento $\phi : T \rightarrow B$.*

Proposição 5 Uma rede neural bem formada $\mathcal{NN} = (E', R')$ é um sistema normal dinâmico.
Demonstração: Uma rede neural bem formada, $\mathcal{NN} = (E', R')$, é um sistema normal (Proposição 4).

Para mostrar que ela é um sistema dinâmico primeiro será definido o conjunto de todas as trajetórias, Ω, sobre o espaço de direções para redes neurais. Depois será mostrado que ela satisfaz os 3 ítens requeridos pela Definição 10.

Ao se trabalhar com redes neurais é suficiente considerar o tempo como discreto para determinar o comportamento da rede. Seja, então, $D_1 = \{t_1, t_2, t_3, \cdots, t_m\}$ um conjunto do tempo linearmente ordenado por $\leq$. Como existe apenas um conjunto de direção, $\mathcal{D} = D_1$. Assim, Ω é o conjunto de todas as trajetórias sobre $\mathcal{D} = \{t_1, t_2, t_3, \cdots, t_m\}$. Então, tem-se:

(a) W, A, $\Theta \in E'$ são conjuntos de variáveis do sistema (Proposição 1).

(b) $F \in R'$ e $F = \{F_W, F_A, F_\Theta\}$ é um conjunto de relações comportamentais globais (Proposição 4).

(c) $F = \{F_W, F_A, F_\Theta\}$ é um conjunto de relações comportamentais globais. Para cada F_i uma relação ρ_i entre Ω e $\mathcal{P}(F_i)$ é definida:

$\rho_W \subset \Omega \times \mathcal{P}(F_W)$ tal que para cada par $[(T, \leq), B_W] \in \rho_W$, existe um mapeamento $\phi_W : T \rightarrow B_W$.

$\rho_A \subset \Omega \times \mathcal{P}(F_A)$ tal que para cada par $[(T, \leq), B_A] \in \rho_A$, existe um mapeamento $\phi_A : T \rightarrow B_A$.

$\rho_\Theta \subset \Omega \times \mathcal{P}(F_\Theta)$ tal que para cada par $[(T, \leq), B_\Theta] \in \rho_\Theta$, existe um mapeamento $\phi_\Theta : T \rightarrow B_\Theta$.

Assim as tres condições para um sistema ser dinâmico foram satisfeitas. Então, uma rede neural bem formada é um sistema normal dinâmico como se pretendia demonstrar.

Ao representar uma rede neural bem formada como um sistema normal dinâmico, $\mathcal{NN} = (E', C, F, \Omega, \rho)$, existem algumas de suas características que podem ser facilmente identificadas.

A relação de conexão C representa a estrutura ou padrão de conectividade da rede. Existem várias propriedades que podem ser extraídas analisando apenas a estrutura da rede. Como por exemplo, determinar o *fan-in* (número de entradas) e o *fan-out* (número de neurônios que serão estimulados por esta saída) para o neurônio n_j; classificar os neurônios em tres tipos: entrada, saída e escondidos; identificar conexões retroativas (feedback); identificar modelos de redes com arquitetura de multicamadas; e outras. A presença ou não de tais propriedades dentro da relação C irá determinar diferentes tipos de arquiteturas.

O fato de uma rede neural ser um sistema normal dinâmico sugere que as variáveis do sistema irão assumir valores diferentes. Qualquer mudança no valor de uma variável resulta em um novo estado e uma nova configuração da rede. O comportamento da rede é, então, analisado através da função ρ, conforme definição de sistema dinâmico.

4 Conclusão

Este artigo apresentou uma proposta de formalização em redes neurais. Partiu-se da análise dos diferentes modelos existentes com objetivo de identificar os componentes e relações que fossem comuns entre eles. Identificados os elementos canônicos, trabalhou-se numa representação matemática que pudesse expressar a natureza e comportamento destes elementos e seus relacionamentos. Resultou disto um modelo geral que foi experimentado na descrição de alguns paradigmas de redes neurais. Este modelo descreve uma rede a nível de neurônio, arquitetura, aprendizagem e modo de operação, apresentando um aspecto temporal. Tendo concebido o modelo, foi apresentada a equivalência das redes neurais bem formadas com sistemas normais dinâmicos. A partir desta equivalência foi possível trazer algumas propriedades encontradas na teoria geral de sistemas para sistemas neurais. A adequabilidade de representar redes neurais bem formadas como sistemas normais dinâmicos fortaleceu o modelo de especificação formal para redes neurais, MOFEU.

É importante ressaltar que a universalidade do modelo proposto ainda nao foi provada. Os exemplos construídos fornecem uma boa espectativa de que o modelo seja universal. Entratanto, para provar tal propriedade é necessário utilizar o modelo na descrição de outros paradigmas de redes neurais e, possivelmente, adaptá-lo mediante algumas necessidades de especificação mais genéricas. A denominação *universal* está presente para destacar a intencionalidade do modelo, o qual foi elaborado tendo em vista uma grande variedade de paradigmas em redes neurais.

Como perspectiva futura para este trabalho está a possibilidade de se ter uma ferramenta genérica de análise do comportamento dos modelos de redes neurais através da sua animação a partir de uma especificação dada. O MOFEU é o ponto de partida para a realização deste objetivo.

References

[1] I Aleksander. Adaptive pattern recognition systems and boltzmann machines: A rapprochement. *Pattern Recognition Letters*, 6:113–120, 1987.

[2] Mario Bunge. *Teoria e Realidade*. Perspectiva, 1974.

[3] Antonio Caselles. Structure and Behavior in General Systems Theory. *Cybernetics and Systems*, 23:549–560, 1992.

[4] E Fiesler and H. John Caulfield. Neural network formalization. Technical report, Institut Dalle Molle d'Intelligence Artificille Perceptive, Suisse, 1992.

[5] E C D B C Filho, D L Bisset, and M C Fairhurst. A goal seeking neuron for Boolean neural networks. In *Proc. International Neural Networks Conference*, volume 2, pages 894–897, Paris, France. July 1990. IEEE.

[6] S Grossberg. Adaptive pattern classification and universal recoding: Ii. feedback, expectation, olfaction, illusions. *Biological Cybernetics*, 23:187–202, 1976.

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

[8] Shun ichi Amari. Mathematical Theory of Neural Learning. *New Generation Computing*, 8:281–294, 1991.

[9] Paul Krause and Andrzej Glowinski. Formal Specifications and Medical Decision Support Systems. *Applied Artificial Intelligence*, 7:237–256, 1993.

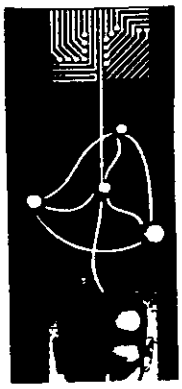
[10] W S McCulloch and W H Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull Math Biophys*, 5:115–133, 1943. formal neuron.

[11] M Minsky and S Papert. *Perceptrons*. MIT Press, Cambridge, 1969.

[12] D. A. Santos. E. C. D. B. C. Filho, M. M. C. Costa, and B. M. Acioly. Um Modelo Formal de Especificação de Redes Neurais. In *XIX Latinoamerican Informatics Conference 22 JAIIO PANEL'93*. volume 3, chapter 13, pages 237–246. Buenos Aires, August 1993.

[13] Débora Abdalla Santos. Um MOdelo Formal de Especificação Universal para Redes Neurais – MOFEU. Master's thesis, Universidade Federal de Pernambuco, Departamento de Informática, 1993.

[14] Débora Abdalla Santos and Edson Costa de B. C. Filho. Equivalência de Redes Neurais Bem Formadas e Sistemas Normais Dinâmicos. A ser publicado nos anais da XX Conferência Latinoamericana de Informática, Atizapan de Zaragoza, Mexico, September 1994.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

A Generalization of Graded Response Formal Neurons

Osame Kinouchi
Instituto de Física, Universidade de São Paulo,
CP 20516, CEP 01452-990 São Paulo, SP,
Brazil
E-mail: osame@if.usp.br
Marcelo H. R. Tragtenberg
Departamento de Física, Universidade Federal
de Santa Catarina,
CP 476, CEP 88049-900 Florianópolis, SC,
Brazil
E-mail: fsc1mht@brufsc.bitnet

Abstract

We introduce a simple generalization of graded response formal neurons which presents very complex behaviour. Phase diagrams in full parameter space are given, showing regions with fixed point, periodic, quasiperiodic and chaotic behaviour. The diagrams also represent the possible time series implementable by the simplest feed-forward network, a two-input single-layer perceptron.

1 Introduction

Although convergent dynamics (relaxation to fixed points) has been a dominant theme in artificial neural networks studies [1], oscillatory behaviour in network models is receiving growing atention. Oscillatory *synchronized* dynamics has been observed at different biological functional levels, for example in the visual cortex [2] and sensoriomotor cortex [3]. Such behaviour has attracted great attention since it seens to be an important cue for solving the feature-binding problem (how the brain links different perceptual signals to the same object).

This has prompted some researchers, which have previously worked with networks of formal neurons (Hopfield attractor networks with MacCulloch-Pitts or graded response elements), to consider with renewed interest models of coupled oscillators [4, 5, 6], since it has not been clear how to extend the previous Hopfield paradigm to collective synchronization phenomena.

Other researchers, from the dynamical systems community, have proposed coupled maps lattices (CML) as an alternative paradigm to consider synchronization, chaos and other spatio-temporal phenomena in biological neural networks [7]. The most studied models use as basic element the logistic map, perhaps because this map is simple and well known.

In this work we propose a new formal neuron which is a n-dimensional map (then, networks of these elements lie in the coupled maps paradigm). However, this map is a very natural extension of the graded response neurons popularized by Hopfield [8]. We call this map a *dynamical perceptron* (DP). A network of totally connected DPs constitutes a globally coupled maps (GCM) system [7] which is a simple generalization of Hopfield networks of graded response neurons [9, 10].

This is a very *preliminar* study where we explore some aspects of the model. In section 2 we introduce the dynamical perceptron map and in section 3 we present phase diagrams in parameter space for the $n = 2$ case. Section 4 contains our conclusions and outlook.

2 The model

Neuron models have two deal with to competing constraints: biological realism and mathematical tractability. Some models (for example, Hodgkin-Huxley neuron) stress the first one and others (formal neurons) achieve the second one only throught dramatic simplifications. Hodgkin-Huxley neurons present a varied repertoire of dynamical behaviours depending on its parameter values, but are not easy to study analytically or even computationally. On the other hand formal neurons most considered in the literature are structurally simple but have no intrinsic dynamics.

We adopt the physicist approach where, for studing the *necessary* (in contrast with the *sufficient*) requisites for the appearance of some collective behaviour in networks, the simplest elements with the simplest interactions should be first considered. But we want to propose a formal neuron which, althought structurally simple, presents complex dynamical behaviour at least *qualitatively* similar to real neurons. Unlike other proposals, however, our model stays within the formal neuron paradigm popularized by Hopfield, since it is a simple generalization of the graded response neuron widely used in the literature.

The two most considered formal neurons are the *McCulloch-Pitts neuron* (a binary variable like an Ising spin) and the *graded response neuron* [8]. The state variable of the graded response neuron is devised to describe not the single action potentials (which are all-or-none events) but the average frequency of these spikes. This average frequency we will call the *activity* $V(t)$ of the neuron. The neuron activity response to external inputs can be approximated by a sigmoidal curve. This is one of the reasons behind the popular choice of the hiperbolic tangent as the formal neuron transfer function [1].

Consider a time series of this average firing activity (how this time series can be obtained is not our concern now). Our task will be to find a simple model which describes such time series. A simple and general approach to this modelling problem, which we will follow here, is to use a feed-forward artificial neural network (FANN) to emulate the system behaviour [1].

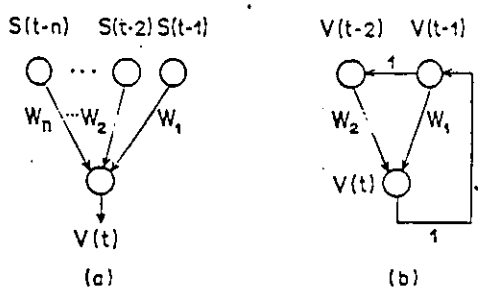


Figure 1: a) single-layer perceptron; b) dynamical perceptron with $n = 2$.

It is well known that FANNs with one layer of hidden units can approximate any continuous function. Then, we can in principle to train a feed-forward net to reproduce the biological neuron time series (a somewhat self-refferential application of neural nets!). Once we find such FANN (which will be a formal neuron model) we can couple various of them to form a large net of such formal neurons. At the end of this process we will achieve a large, complicated and theoretically intractable system.

Well, let's simplify our task. We ask for *the simplest feedforward net which qualitatively models the time series.* By *qualitatively* we mean a very weak requeriment: we want a formal neuron which presents, like biological neurons, not only graded response but also oscillatory behaviour in certain circumstances.

A single-layer perceptron (a feedforward net without hidden units) can do this and has the advantage of being a very well studied system (see figure 1a). The perceptron learns the time series $\{S(t)\}$ by using *examples*, that is, input-output pairs $(\mathbf{X}_t, Y_t)$. The desired output is the value of the time series at time t, $Y_t \equiv S(t)$. The input may be the previous n values of the time series $\mathbf{X}_t \equiv \{S(t-1), \ldots, S(t-n)\}$. Then, the examples are generated simply by slipping a window of lenght n over the data. The perceptron output will be given by [1]

$$\begin{aligned} V(t) &= g\left(\mathbf{W}.\mathbf{X}_t - \theta\right) \\ &= g\left(\sum_{\tau=1}^{n} W_\tau S(t-\tau) - \theta\right) \end{aligned} \qquad (1)$$

where $\mathbf{W} = \{W_1, \ldots, W_n\}$ is the weight vector

which defines the perceptron, the constant θ is the so called *bias* term and $g(x)$ is a non-linear continuous sigmoidal function.

After the trainning over a time series (remember that no perfect reprodution is required) we can create a dynamical model from the perceptron so obtained (we will call this system a *dynamical perceptron*, see figure 1b). To do this we simply send the perceptron output to the last input node and transfer the value present at each input node to its left neighbour (the extreme left value being discarded). That is, the dynamical perceptron output will be given by the following n-order recurrence equation

$$V(t) = g\left(\sum_{\tau=1}^{n} W_\tau V(t-\tau) - \theta\right) \qquad (2)$$

We will specialize the transfer functions $g(x)$ to the family of sigmoidal functions

$$g(x) = c_1 + c_2 \tanh(\gamma x) \qquad (3)$$

where c_1 and c_2 are constants. The usual sigmoid with image in the $[0,1]$ interval is obtained with $c_1 = c_2 = 1/2$. The hiperbolic tangent, of course, is given by $c_1 = 0$ and $c_2 = 1$. All these maps are topologicaly conjugated to the tanh one throught a change of variables $V' = (V - c_1)/c_2$ giving

$$V'(t) = \tanh\left[\gamma'\left(\sum_{\tau=1}^{n} W_\tau V'(t-\tau) - \theta'\right)\right] \qquad (4)$$

with the transformed variables

$$\gamma' = c_2\gamma, \qquad \theta' = \frac{\theta}{c_2} - \frac{c_1}{c_2}\sum_{\tau=1}^{n} W_\tau \qquad (5)$$

We will concentrate our attention in the *simplest* non-trivial dynamical perceptron, that is, the one with $n = 2$. We also note that a dynamical perceptron with $n = 1$ has been proposed as a formal neuron by Pasemann [11]. However, the $n = 1$ model has very poor behaviour, exhibiting only fixed points or two-cycle as attractors. Networks of such neurons have been considered by statistical physiscists [9].

The 2-D map, instead, presents very rich behaviour (fixed points, limit cycles, quasiperiodic and chaotic attractors, coexisting attractors etc.) which can be used to mimic some known behaviours of real neurons. It also allows the modelling of time-dependent neuron responses (habituation, sensitization) because the internal parameters $\mathbf{W}$, and not only the external synapses, can suffer a learning process.

We write the recurrence equation for the 2-dimensional DP as

$$V(t) = \tanh\left[\frac{V(t-1) - \kappa\, V(t-2) + H}{T}\right] \qquad (6)$$

where

$$T = \frac{1}{\gamma' W_1}, \quad \kappa = \frac{-W_2}{W_1}, \quad H = I - \theta', \qquad (7)$$

where we have allowed the possibility of an external input I and γ' and θ' are given by eq. (5). This is the simplest neural network model of a dynamical system and the full phase diagram in the variables (T, κ, H) gives all the possible time series implementable by this architecture.

The choice of these variables is motivated by convenience in the visualization of the phase diagram as well to connect the 2-D dynamical perceptron with previous literature. This map (for $H = 0$) has been studied in the contex of statistical mechanics models of magnetically modulated materials by Yokoi, Oliveira and Salinas [12], where $V(t)$ describes the magnetization on the t^{th} layer of an Ising model with competing interactions in a Bethe lattice. Recently, Yokoi and one of us (MHRT) have studied the map for constant and non-zero H [13].

3 Phase Diagrams

The first diagram (fig. 2, 3) refers to the dynamical behaviours in the plane T vs κ for $g(x) = tanh(x)$. The main periodic phases are shown and their labels $q/2\pi = P/Q$ denote their periods Q (the number of points after which the series repeats) and the number of crests P within a period (q is the 'modulated phase wave number'). Quasiperiodic phases have $q/2\pi$ irrational.

In the region 0 there are only trivial fixed points. In the region 1 two fixed points coexist for all values of T and κ, and the neuron states are the solutions of $V = \tanh((1-\kappa)V/T)$. The convergence to each fixed point depends on the initial conditions $V(t=0)$ and $V(t=1)$. Phase 1/2 is

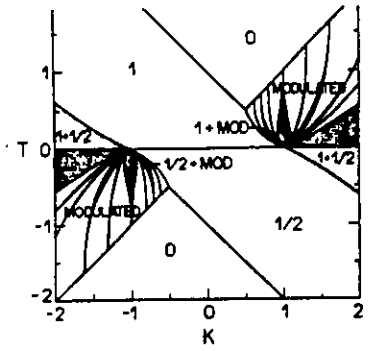


Figure 2: Global phase diagram in the $(T, \kappa, 0)$ plane.

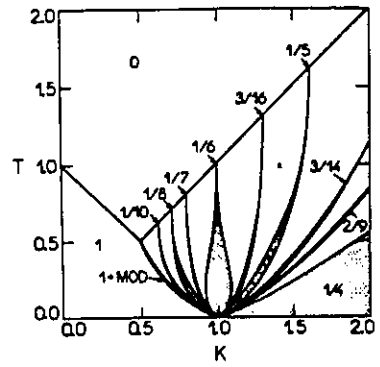


Figure 3: Main periodic phases in the first quadrant. The diagram on the third quadrant is symmetric to this one, with a serie $(V_1, V_2, V_3, V_4 \ldots)$ being changed to $(V_1, -V_2, V_3, -V_4, \ldots)$.

a two-cycles and the more complicated phase 2/9 is depicted as an example (fig. 4). Quasiperiodic phases and other periodic phases lie between the grey regions. Details concerning the determination of these lines are given in [13].

In the regions 1 + 1/2 a fixed point and a two-cycle coexists. More interesting is the modulated-fixed point coexistence region (1+MOD) where periodic, quasiperiodic and chaotic attractors coexists [12, 13]. The same occurs at the coexistence region 1/2+MOD.

In figures 5, 6 and 7 we show diagrams in the

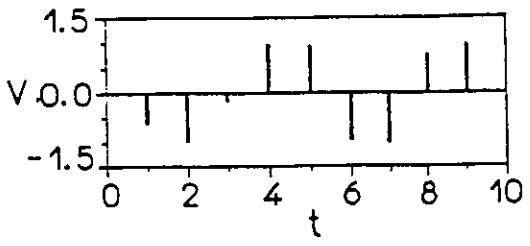


Figure 4: Example of a 2/9 phase. The series repeat each interval of 9 points, but there are 2 oscillations in the interval. The 'average wave lenght' is $\lambda = 9/2$.

plane T vs H, which are the most interesting for our neuron modelling purpose since they give the neuron response to external (constant) input I. The stability lines for the 0-phase are given by

$$H_c^{\pm} = (p-1)M_c^{\pm} + T \tanh^{-1} M_c^{\pm} \qquad (8)$$

with

$$M_c^{\pm} = \pm\sqrt{1 - \frac{T}{1-\kappa}}, \quad (p < 1/2) \qquad (9)$$

$$M_c^{\pm} = \pm\sqrt{1 - \frac{T}{\kappa}}, \quad (p > 1/2) \qquad (10)$$

Five general types of behaviour are encountered in these diagrams. For example, a neuron with sigmoid transfer function ($c_1 = c_2 = 1/2$) and delayed self-coupling $\kappa = 0.7$ (fig.6), with gain and threshold situated at point A will represent a formal neuron with resting near zero activity which, under external input, will show graded response. Neuron B shows, at intermediate external input level, oscillatory behaviour. Neuron C is a natural oscillator, a pacemaker with periodic or quasiperiodic behaviour even without external input. Neuron D has a very interesting 'bistable' behaviour. It has two coexisting attractors, one with low activity and other with high activity. An external instantaneous perturbation can put the neuron in the active state (figure 8). Neuron E is bistable at intermediate levels of external input I. Behaviours of type B and D remember some findings on thalamic neurons [14].

Detailed diagrams for the modulated regions with different values of κ are given in ref. [13]. As an example of the complex nature of these regions we show the diagram for $\kappa = 1$ (figure 9). In the coexistence region 0+MOD (inferior region of the bubble) we can also find chaotic attractors like the presented in figure 10.

4 Outlook and Conclusions

How does a network of DPs relates to the attractor (Hopfield-like) networks studied by the statistical physicists? Well, the most studied network [9, 10] is a fully connected set of N formal neurons without self-couplings and thresholds, with the (parallel) neuron dynamics given

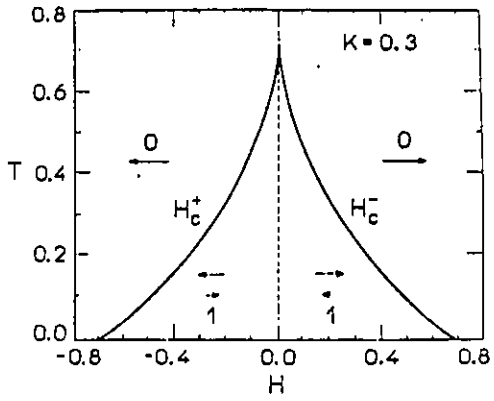


Figure 5: Stability limits for the 0 phase for $\kappa = 0.3$. In the region 1 coexists a phase parallel to H and other with smaller amplitude antiparallel to H.

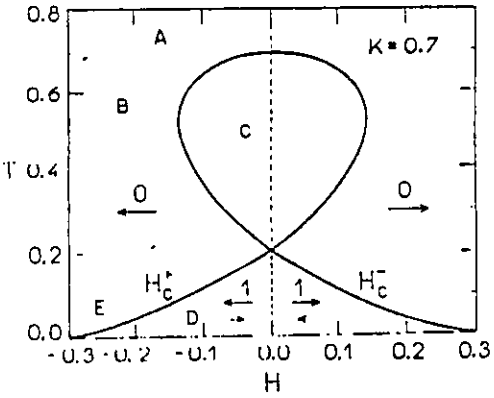


Figure 6: $\kappa = 0.7$: Modulated phases appear inside the bubble. A, B, C, D and E represent neuron models with qualitatively different dynamical responses.

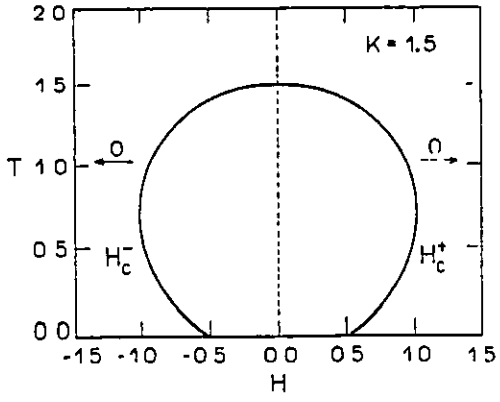


Figure 7: $\kappa = 1.5$. Modulated phases appear inside the bubble. There is no phase 1 for $\kappa > 1$.

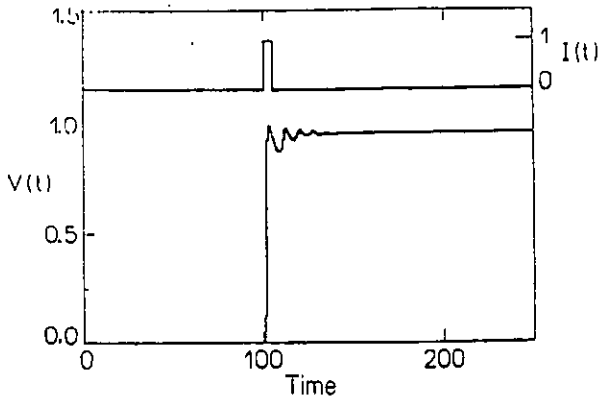


Figure 8: Plots of the activity $V(t)$ and external input $I(t)$ for a neuron of type D.

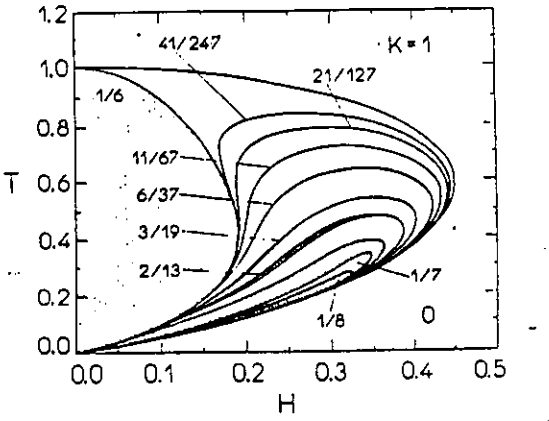


Figure 9: T vs H phase diagram for $\kappa = 1.0$

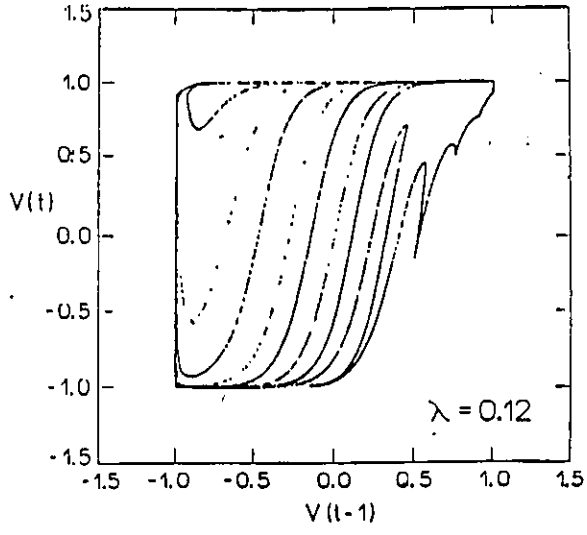


Figure 10: Strange attractor with largest Lyapunov exponent $\lambda_1 = 0.12$ found at the point $(T = 0.15, \kappa = 1, H = 0.235)$.

by

$$V_i(t) = \tanh\left[\gamma \sum_{j=1}^{N} J_{ij} V_j(t-1)\right] \qquad (11)$$

with $J_{ii} = 0$. A phase diagram showing the recuperation (content-adressable memory), paramagnetic and spin glass phases has been obtained by Shiino e Fukai [10] using the replica method. This phase diagram is very similar to the obtained by Amit et. al. [1] for the Hopfield network composed of McCulloch-Pitts neurons. The network presents only fixed points to be indentified with stored memories.

A network with self-couplings $J_{ii} = W_{1i}$ has also been considered, the neuron dynamics given by

$$V_i(t) = tanh\left[\gamma\left(\sum_{j=1,j\neq i}^{N} J_{ij} V_j(t-1) + W_{1i} V_i(t-1)\right)\right] \qquad (12)$$

A condition of stability for the fixed point phase was determined by Marcus and Westervelt [9]. If this condition is not satisfied a limit cycle of period two appear.

A network of 2-D dynamical perceptrons will have the dynamics given by

$$V_i(t) = tanh\left[\gamma\left(\sum_{j=1,j\neq i}^{N} J_{ij} V_j(t-1) + W_{1i} V_i(t-1) + W_{2i} V_i(t-2)\right)\right] \qquad (13)$$

This network cannot be analysed by the usual statistical mechanics approach, being a globally coupled maps system of the type studied (almost numerically) by the dynamical systems comunity. The 'easy' ferromagnetic case with all $J_{ij} = J > 0$ should be studied first, where global synchronization phenomena may appear. It seems also interesting to study a system with well chosen couplings (say Hebb synapses) designed to store various synchronized attractors. The analysis of such systems is a future enterprise.

In conclusion, the phase diagrams in the (T, κ, H)–space represent all the time series implementable by a two input single-layer perceptron. We think that the richness of dynamical behaviours encountered has pedagogical value since gives us some feeling about the much more varied time series implementable by more complex networks. To our knowledge these are the first phase diagrams in coupling space for the behaviour of a feedforward neural network turned a dynamical system.

Our proposal is to represent neurons by these two-parameter *dynamical perceptrons*. By this slight extension of the graded response formal neuron we can model qualitatively various dynamical behaviours of biological neurons. Networks of such discret oscillator neurons enables the study of collective synchronization phenomena, which will be reported elsewhere.

References

[1] Hertz J. A., Krogh A. and Palmer R. G., 'Introduction to the Theory of Neural Computation' (Addison-Wesley, Redwood City, CA) 1991.

[2] Gray C. M., Konig P., Engel A. K. and Singer W., *Nature* **338** (1989) 334.

[3] Murthi V. N. and Fetz E. E., *Proc. Natl. Acad. Sci. USA* **89** (1991) 5670.

[4] Abbott L.F., *J. Phys. A: Math. Gen.* **23** (1990) 3835.

[5] Sompolinsky H., Golomb D. and Kleinfeld D., *Phys. Rev. A* **43** (1991) 6990.

[6] Hansel D. and Sompolinsky H., *Phys. Rev. Lett.* **68** (1992) 718.

[7] Kaneko K., *Chaos* **2**(3) (1992) 279.

[8] Hopfield J. J., *Proc. Natl. Acad. Sci. USA* **81** (1984) 3088.

[9] Marcus C. M. and Westervelt R. M., *Phys. Rev. A* **40** (1989) 1.

[10] Shiino M. and Fukai T., *J. Phys. A: Math. Gen.* **23** (1990) L1009.

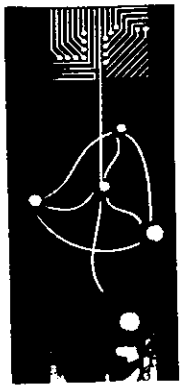
[11] Pasemann F., *Int. J. Bifurc. Chaos* **3** (1993) 271.

[12] Yokoi C. S. O., de Oliveira M. J. and Salinas S. R., *Phys Rev. Lett.* **54** (1985) 163.

[13] Tragtenberg M. H. R. and Yokoi C. S. O., IFUSP-preprint.

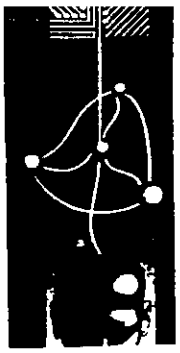
[14] Rose R. M. and Hindmarsh J. L., *Proc. R. Soc. Lond.* **B25** (1985) 161.

Anotações



NOVAS ARQUITETURAS

Anotações



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajuba
Itajuba, 24 a 27 de outubro de 1994

Hybrid Networks: A Selective Committee Architecture

Antonio G. Thomé
Instituto Militar de Engenharia
Departamento de Engenharia Eletrica
Rio de Janeiro - Brasil
e-mail: s3thome@imerj.bitnet

Manoel F. Tenório
Purdue University
School of Electrical Engineering
Indiana - USA
tenorio@ecn.purdue.edu

Abstract. In this report we describe a hybrid network technique to generate a committee architecture for a time series prediction case study. The algorithm, here named Selective Multiple Prediction Network - SMP, consists of three steps: a systematic partition of the input hyperspace, a selective training of many agents and a flexible combining strategy. This algorithm generates potentially uncorrelated agents which may improve the performance of the combination process. The proposed architecture is easily extended to the class of pattern classification problems.

Key words: Committe Architecture, Team Prediction, Hybrid Architecture

1. Introduction

System Identification and Linear Prediction are two very important topics in the field of System Theory, and are widely applied to many diverse areas such as Signal Processing, Control, and Forecasting. System Identification is the process of estimating an unknown structure by the knowledge of only its input / output pairs. Linear Prediction, on the other hand, is the process of estimating a future system response based only on the knowledge of its present and past responses.

The emphasis on creating a predictor relies on the identification of underlying patterns, and on the estimation of the model parameters. Historical data analysis and pattern consistency are respectively the major resource and the major underlying assumption for the estimation of such parameters. It may be intuitive the understanding that more complex the problem the more difficult is the underlying patterns identification, and even more difficult is the estimation of a single model that satisfactorily covers the entire problem.

The main idea here is that a large class of prediction problems can be better solved by decomposing the original problem into several subproblems and then combining the multiple sub-solutions, something like divide-and-conquer. This approach generally leads to simpler individual networks and also to a higher accuracy than solving the problem with a single and global predictor. Our committee approach has a simple architecture (fig. 1) composed of three distinct modules: the selector, that performs a pattern classification; the predictor, that is a set of simple networks working in parallel; and the combiner, that generates the system output. The SMP training algorithm consists basically of three steps as follows

- decomposition of the original problem into several and ideally disjoint subproblems;
- parallel estimation of the parameters for each model (agent); and
- combination strategies.

The first step, partition of the original problem, relies on the application of unsupervised methods such as K-means and fuzzy locally sensitive [Tho93a] clustering algorithms. For time series problems, for example, it turns out to be necessary a previous transformation to the original time dependent sequence of points generating a set of state-space vectors and then, spatial similarities of these vectors are exploited by the use of an unsupervised clustering procedure.

In the second step, as many agents as the numbers of partitions are trained, in parallel, on the subsets created by the clustering procedure. Linked to each cluster there is a corresponding agent that can be seen as an expert on a particular view of the underlying structure. Uncorrelated agents are expected to result from this training scheme. Each agent provides its own prediction, and the network or committee final prediction is then obtained through the combination of the individual contributions. We propose here three different combining strategies.

2. The Combining Paradigm

In a seminal paper [Bat69], Bates and Granger showed that a simple linear combination of distinct predictions generally outperforms the individual predictions. A stream of papers followed this initial work. Clemen and Winkler [Cle86] and Clemen [Cle89] provide excellent summaries and extensive bibliographic references.

The field has so far been dominated by works in statistical decision theory, with particular emphasis

on optimal linear combination and on Bayesian inference. However, connectionist researchers have recently begun to show a strong interest in the subject in the form of network committees, agent teams, stacked generalization and others [Lit91, Bey93, Wol92, Sch89, Mac93, Zha92].

Combining is theoretically no worse than any of the individual agents, which can be shown as follows:

Let A_α and A_β be two distinct agents working on information sets I_α and I_β, and let f_α^n and f_β^n be their corresponding predictions for time step n.

If the predictions are optimal with respect to their respective information sets they can be written in terms of posterior expected values, i.e.

$$f_\alpha^n = E\{X_n / I_\alpha\}, \quad (1)$$

and

$$f_\beta^n = E\{X_n / I_\beta\}. \quad (2)$$

The optimal prediction, based on all possible information is then known to be

$$f^n = E\{X_n / I_\Gamma\}, \quad I_\Gamma = I_\alpha \cup I_\beta. \quad (3)$$

This complete estimation problem is normally very complex and computationally expensive. A particular subset of $\{I_\Gamma\}$ that can be considered for example, is a linear combination of the individual predictions

$$C^n = \alpha_1 f_\alpha^n + \alpha_2 f_\beta^n. \quad (4)$$

It is expected that α_1 or α_2 should go to zero whenever f_α^n or f_β^n is optimal with respect to the global information set (I_Γ). If neither one is optimal then α_1 and α_2 are expected to be different from zero. In general, C^n and f^n are not equal, which clearly indicates that the combination will not be optimal too, although a superior result to each of the original predictions is expected.

Although showing potential for performance improvement, combining techniques present some weak points. The combined performance is highly dependent on the estimation error cross-correlation, serial correlation, and bias. The most effective combinations are achieved with no positive cross-correlation between individual model errors. When negative correlation occurs, which is quite rare, the gains can be spectacular. However, with high positive cross-correlation it is often difficult to achieve even a small improvement. Moreover, if an unstable optimization technique is used, the results may be even worse than those of using equal weights or of selecting the apparently best model. Therefore, the keystone for any combining scheme relies on the generation of as less correlated agents as possible.

3. The Selective Multiple Prediction Network

Training agents over distinct subsets of the full training set is not a new idea. Wolpert [Wol92] uses arbitrarily selected partitions to train the first layer of generalizers; Schapire [Sch89] adopts a residual scheme in which every new agent is trained only on those vectors which previous agents have disagreed on. Ersoy's parallel, self-organizing, hierarchical neural networks [Ers89, Hong91], can also be seen as a kind of residual partition where new agents are trained on transformed versions of those samples rejected by previous agents. In SMP a different scheme to partition the input data set and to perform the prediction task is used. First of all a clustering algorithm is adopted to subdivide the original problem into sets of more homogeneous and easier subproblems, which may eventually lead to learning and prediction improvements, and later, many agents are used in parallel to provide the network output.

The Selective Multiple Prediction Network (fig. 1) involves three distinct processing steps and a number of distinct agents working in parallel. These agents can form a hybrid or a homogeneous structure depending on how they differ from one another. In our studies we only considered homogeneous systems in which each agent is a neural back-propagation network.

3.1 Processing Steps

Pattern matching, function approximation, and a combining strategy are the most important components of the selective multiple prediction task. Pattern matching involves feature selection and unsupervised learning; function approximation involves selective supervised learning, where each neural network is trained to become an expert on specific views of the entire environment; and the combining strategy generates the final prediction. Figure 2 shows a block diagram of these steps.

The selection of relevant features is the first and one of the most important steps. In univariate time series problems this selection process can be thought of in terms of defining different embedding dimensions, i.e., the number of past values to be used in the model. Feature selection [The89, Hsu93, Lap86] is generally a very time consuming and complex task. Here we favored the use of a spread ratio measure (s) (eqn 6) to select those possible embeddings leading to a more consistent unsupervised partition of the input space. The model for a time series is generally expressed as

$$Y = f(X) + \varepsilon, \quad (5)$$

where

$X = [x(t)\ x(t-\tau)\ x(t-2\tau)\ \ldots\ x(t-(m-1)\tau)]^T$ is a vector of mx1,
$Y = x(t+T)$ is a scalar value,
τ is the sampling period,
m is the embedding size,
T is the prediction horizon (lead time).

The spread ratio measure is defined as

$$r_s = mean\left(r_\sigma^i r_f^i\right) \qquad (6)$$

where

$$r_\sigma^i = \frac{mean\left(\sigma_i^2\right)}{\sigma^2},$$

measures the data consistency,

σ_i^2 is the outcome variance for the input vectors belonging to cluster i

σ^2 is the outcome variance for the whole training vectors, and

$$r_f^i = \frac{mean\left(d_w^i\right)}{mean\left(d_B^i\right)}, \qquad (7)$$

is the Fisher descriminator term, which measures the ratio within (d_w) x between (d_B) cluster distances

$$d_w^i = \frac{1}{n_i}\sum_{j=1}^{n_i}\left\|X_j - V_i\right\|^2, \quad \left\{\forall j / X_j \in Cluster\ i\right\}$$

$$d_B^i = \frac{1}{c-1}\sum_{j\neq i}\left\|V_j - V_i\right\|^2,$$

where V_i represents the center of mass of the i^{th} cluster.
Once the embedding m is determined, the time series can be rewritten as a collection of input vectors X, also known as state vectors, and their corresponding outcome Y. This performs a transformation from time to spatial domain where the time dependence is respected within each vector but ignored among different vectors; the regression problem can then be viewed as a case of pattern association of pairs of vectors as follows

x_{11}	x_{12}	.....	x_{1n}
x_{21}	x_{22}	.....	x_{2n}
:	:	:	:
:	:	:	:
x_{m1}	x_{m2}	.....	x_{mn}
y_1	y_2	.....	y_n

This transformation is the key to instance-based methods [Aka91, Hsu93] in which the outcome prediction for the current state vector is based on the outcomes of a number of past state vectors found using a look-up search and a kind of k nearest neighbor approach

Here, in the SMP algorithm, a similar transformation is applied to the time series with the objective to subdivide the original problem into a set of more homogeneous subproblems. An embedding size is selected and the original time series is then transformed into a set of state vectors of that chosen dimension. To select the optimal embedding we used the fuzzy locally sensitive clustering, described in [Tho93a], and also a K-means algorithm. The previously mentioned performance criterion (eqn 6) was then applied to identify those embeddings resulting in more consistent and better shaped partitions. The cluster centroids of the best partition were then assumed to be the class representatives for the pattern matching and selective function approximation phases of the SMP algorithm.

Each cluster defines an associated agent that is trained only on those samples that are classified to the corresponding partition. Three strategies for learning and combining the individual predictions to form the committee prediction were evaluated. The first and simplest one, named *winner-take-all*, selects a single agent at every time step to perform the prediction. The selected agent is the one whose corresponding cluster centroid is closest to the current input vector. The second approach, called *full committee*, is at the other extreme, where all agents are taken into consideration. Each agent contributes and is trained on a percentage of the final prediction error. The percentages or weights add up to one and direct correspond to the degree of membership of the current input vector with respect to each cluster. The third approach, called *windowed-committee*, is in between the two others, since it takes into consideration a subset of the available agents. A temporal window is used as a selection criterion to induce time continuity or time similarity as well as spatial similarity. The combination of spatial and temporal similarities has special appeal in time series applications.

4.2 - Learning Procedures

The Quickprop algorithm [Fah88] with adaptive region of nonlinearity, as described in [Tho93b], was used in all experiments. All training data sets were 1500 or more samples long, and the agents (backpropagation networks) were trained in parallel accordingly to each combining strategy. Batch training with a fixed number of epochs upper bounded at 1000, was used. All procedures assume a previous unsupervised partition step where clusters representing the underlying patterns are generated.

case a) Winner-take-all procedure

1 For every state vector in the training set

step 1- classify current input vector with respect to the existing clusters;

step 2- select winning agent (closest one to the current input vector);

step 3- estimate desired outcome; and

step 4- train selected agent through error backpropagation.

2. If desired accuracy is achieved stop else go back to 1.

case b) Full-committee procedure

1. For every state vector in the training set:

step 1- compute degree of membership for current input vector

$$\mu_i = \frac{\exp\left(-\sqrt{(X-V_i)^T(X-V_i)}\right)}{\sum_{i=1}^{c}\exp\left(-\sqrt{(X-V_i)^T(X-V_i)}\right)};$$

step 2- estimate the outcome for all agents in parallel

$$\hat{y}_i = f(X, W_i); \quad i = 1,\ldots,c$$

step 3- generate the committee prediction by combining the individual outcomes

$$\hat{y} = \sum_{i=1}^{c} \mu_i \hat{y}_i;$$

step 4- train each agent by backpropagating its contribution to the overall error

$$e_T = y_d - \hat{y}, \quad and$$

$$e_i = \mu_i e_T, \quad i = 1,\ldots,c.$$

2. If desired accuracy is achieved stop else go back to 1.

case c) windowed-committee procedure

1. For every state vector in the training set:

step 1- classify current input vector

step 2- select winning agent (closest one to the current vector)

step 3- insert the winner vector at the head of the time-window queue (FIFO) and eliminate the oldest entry

$$WD = \left[X_t\, X_{t-1}, \cdots X_{t-k+1}\right],$$

window of size k

where X_t means the winner vector at time instant t.

step 4- estimate the output for each agent belonging to the time-window queue

$$\hat{y}_i = f(WD_i, W_i), \quad i = 1,\ldots,k$$

where WD_i is the i^{th} column vector and W_i is the corresponding set of weight parameters.

step 5- combine individual outcomes

$$\hat{y} = \sum_{i=1}^{k} \lambda_i \hat{y}_i,$$

where

$$\lambda_i = \frac{\beta^i}{\sum_{i=1}^{k}\beta^i}, \quad 0 < \beta \le 1$$

and

$$\sum_{i=1}^{k} \lambda_i = 1, \qquad \text{is the time weight decay}$$

step 6- train each agent by backpropagating its contribution to the overall error

$$e_T = y_d - \hat{y} \quad and \quad e_i = \lambda_i e_T.$$

2. If desired accuracy is achieved stop else go back to 1.

4.3 - SMP Properties and Drawbacks

SMP provides a powerful architecture to deal with complex real world problems. A set of specialized networks are used, rather than a single one which must accommodate all aspects and underlying dynamics of the problem. Specialization, team cooperation, and truly parallel operation are the key issues in SMP. Robustness, complexity and learning effort reduction, and prediction accuracy improvement are the major goals.

Major properties:

- transforms complex problems into a set of more homogeneous and easily treated subproblems;
- uses smaller individual networks which reduces dimensionality problems and improves learning time;
- exploits spatial and temporal similarity of the input vector which is intuitive and appealing for many real world time series applications;
- combines instance based with parametric approaches without the memory and recall time overhead of the former;
- adopts either hybrid or homogeneous structure, with a flexible combining strategy;
- generates potentially distinct agents by training them on different partitions of the training set.

Drawbacks:

- requires large training sets to avoid situations where an agent is trained on a very small number of patterns;
- requires frequent full retraining and input space partitioning if applied to non-stationary time series;
- since each agent has its own distinct training set, which may have different sizes and degrees of complexity, the algorithm may present overfitting problems if the number of training epochs is set equal for all agents and

5. Empirical Results

The Mackey-Glass chaotic time series was chosen for this benchmark due to its common use among connectionist researchers. The purpose behind the use of Mackey-Glass time series was not to show improvements of current estimates that are already at practical limits. Further improvement is of little practical value. Rather, our purpose was to use a chaotic system defined by a continuous orbit, which by nature does not have clearly definable clusters in the state space. If a reasonable prediction can be attained with this thecnique, functions that are clearly decomposable into multiple mappings can therefore, more easily be dealt with.

The Mackey-Glass equation was first proposed as a model of white blood cell production [Mac77] and subsequently popularized in the nonlinear field due to its richness in structure [Far82]. It is a time-delayed differential equation stated as follows:

$$\frac{dx}{dt} = \frac{ax(t-\Delta)}{\left[1+x^c(t-\Delta)\right]} - bx(t) \qquad (7)$$

Which in discrete time domain can be rewritten as:

$$x(t+1) = \frac{ax(t-\Delta)}{\left[1+x^c(t-\Delta)\right]} - (b-1)x(t). \quad (8)$$

In Mackey-Glass benchmarks, it is commonly avoided to draw conclusions based solely on direct numerical comparisons with other published results. This is because of the differences that can arise from the use of different integrators, initial conditions, sampling rate, and transient elimination. In our study, all results are reported in terms of *Nrmse*.

According to Takens, a chaotic time series $x(t)$ can be predicted T time steps in the future by using only m number of equally spaced past samples of the time series itself. The prediction value is then obtained as follows:

$$x(t+T) = F\left\{x(t), x(t-\tau), \ldots, x(t-(m-1)\tau)\right\} \qquad (9)$$

where F, under suitable assumptions, is a nonlinear continuous function. The choice of an embedding scheme for a benchmark means the determination of the three parameters T, m and τ for the time series. In our experiments we adopted the most widely used values, i.e. m= 6, τ= *6 and T= 6 and 85.*

Using the above parameters, many distinct partitions of a training set with 700 samples were evaluated. A K-means clustering algorithm was used for several values of c (number of clusters), and the performance criterion r_s (eqn 6) was evaluated for each resulting partition. The results indicated a systematic partition improvement as the number of clusters increase. Other observation was that the quality of the partition deteriorates as the lead time T goes further in the future. This is because of the chaotic nature of the series.

Winner-take-all, full-committee and windowed-committee schemes were evaluated on different partition sizes for lead times of 6 and 85. Each model (neural network structure) was defined with a single hidden layer (5 units for the T=6 case and 7 units for the T=85 case) and one output unit, hyperbolic tangent with ARON was adopted for all units. Tables 1 and 2 show some of the obtained results.

The architecture of SMP provides the flexibility to customize and individually tune each Agent. Therefore, in this experiment for example, Agents with poor training performance could, in the WTA scheme, be selected for individual retraining and final accuracy may eventually improve.

Table 3 and figure 3 show the committee prediction results for a partition size of 23 clusters. Observe that the prediction provided by the WTA scheme shows very good performance on turning points, with almost no lag, which may be of great interest for some real world applications.

6. Conclusion

The Selective Multiple Prediction Network provides a very flexible and powerful architecture to handle those more complex problems, where a single and global model is very unlikely to exist. Decomposing the original problem into more homogeneous subproblems leads to potentialy uncorrelated and simpler Agents. Less demanding training effort, and customized tuning according to the requirements of each subproblem are some of the characteristics of this approach. This proposed architecture can also be seen as a structure to combine neural networks (prediction module) with more sophisticated schemes of expert systems (selection module)

7. References

[Bat69] Bates, J. M. and Granger, C.W.J., 1969, "The combination of forecasts", Opl Res Q., vol 20, pp. 451-468.

[Bey93] Beyer, U. and Smieja, F., 1993, "Learning from examples, agent teams and the concept of reflection"

[Cle86] Clemen, R. T., 1986, "Linear constraints and the efficiency of combined forecasts", Journal of Forecasting, vol 5, pp. 31-38.

[Cle89] Clemen, R. T., 1989, "Combining forecasts: a review and annotated bibliography", International Journal of Forecasting, vol 5, pp. 559-583.

[Ers89] Ersoy, O. K. and Hong, D., 1989, "Parallel, self-organizing hierarchical neural networks",

Technical Report - TR-EE-89-56. School of Electrical Engineering, Purdue University,IN.
[Fah88] Fahlman, S. E., 1988. "An empirical study of learning speed in back-propagation networks. TR, CMU-CS-88-162.
[Far82] Farmer, D., 1982. "Chaotic attractors of an infinite-dimensional dynamical system", Physica, vol 4D, pp. 366-393.
[Hon91] Hong, D. and Ersoy,-O. K., 1991. "Parallel, self-organizing neural networks", Technical Report - TR-EE-91-13. School of Electrical Engineering, Purdue University,IN
[Hsu93] Hsu, W., 1993. "Nonlinear and self-adapting methods for prediction", Ph.D. Thesis, School of Electrical Engineering, Purdue University, IN.
[Lap87] Lapedes, A. and Farber, R., 1987, "How neural nets work". Proc of IEEE. Denver Conference on Neural Nets.
[Lit91] Littlestone, N. and Warmuth, M.K., 1991, "The weighted majority algorithm", TR UCSC-CRL-91-28. University of California, Santa Cruz, CA.
[Mac77] Mackey, M.C. and Glass, L., 1977, "Oscillation and chaos in physiological control systems", Science, pp. 197-287.
[Mac93] Mackay, D., 1993. "Bayesian non-linear modeling of the energy prediction competition", University of Cambridge, Cambridge, United Kingdon.
[The89] Theilen, C., 1989. Decision Estimate and Classification, John Willey & Sons, NY.
[Tho93a] Thorne, A.G. and Tenorio, M.F., 1993, "A fuzzy locally sensitive method for cluster analysis", Submited to IEEE Transactions on Fuzzy Systems.
[Tho93b]Thorne, A.G. and Tenorio, M.F., 1993, "Accelerated Learning through a Dynamic Adaptation of the Error Surface", Submited to NN magasine.
[Win83] Winkler, R.L. and Makridakis, S., 1983, "The combination of forecasts", Journal of the Royal Statistical Society, Series A, 146, pp. 150-157.
[Wol92] Wolpert, D.H., 1992, "Stacked generalization", Neural Networks, vol 5, pp. 241-259.
[Zha92] Zhang, X., Mesirov, J.P and Waltz, D.L., 1992, "Hybrid system for protein secondary structure prediction", Journal of Molecular Biology.

Committee	Num. Clusters	T=o	T=85
WTA	09	.1962	.4309
WTA	23	.0773	.3403
Windowed	09	.1904	.4233
Windowed	23	.0728	.3194
Full-Committee	23	.1221	.3752

Table 2 - Mackey-Glass Committee Training Performance in Nrmse

Cls	No	Ns	Cls	No	Ns	Cls	No	Ns
01	.1182	.1117	09	.2285	.2742	17	.1551	.2677
02	.5304	.3842	10	.6292	.6807	18	.3656	.3961
03	.2167	.2912	11	.0660	.0712	19	.5488	.6979
04	.7403	.1614	12	.7220	.6555	20	.3548	.4264
05	.1727	.1478	13	.2011	.2296	21	.5299	.4234
06	.3123	.1949	14	.2516	.1834	22	.2752	.2823
07	.3374	.3378	15	.3698	.3582	23	.5284	.3587
08	.2821	.2876	16	.1447	.1548	WTA	.3403	.3694

Table 3 - Mackey-Glass WTA-Committee Training/Prediction for T=85 (Cls - cluster number, No - cluster training Nrmse, Ns - cluster testing Nrmse; partion size of 23 clusters)

Committee	T=o	T=85
WTA	.0825	3694
Windowed	.0835	3653
Full	.1123	

Table 4 - Mackey-Glass Committee Prediction Comparison [illegible]

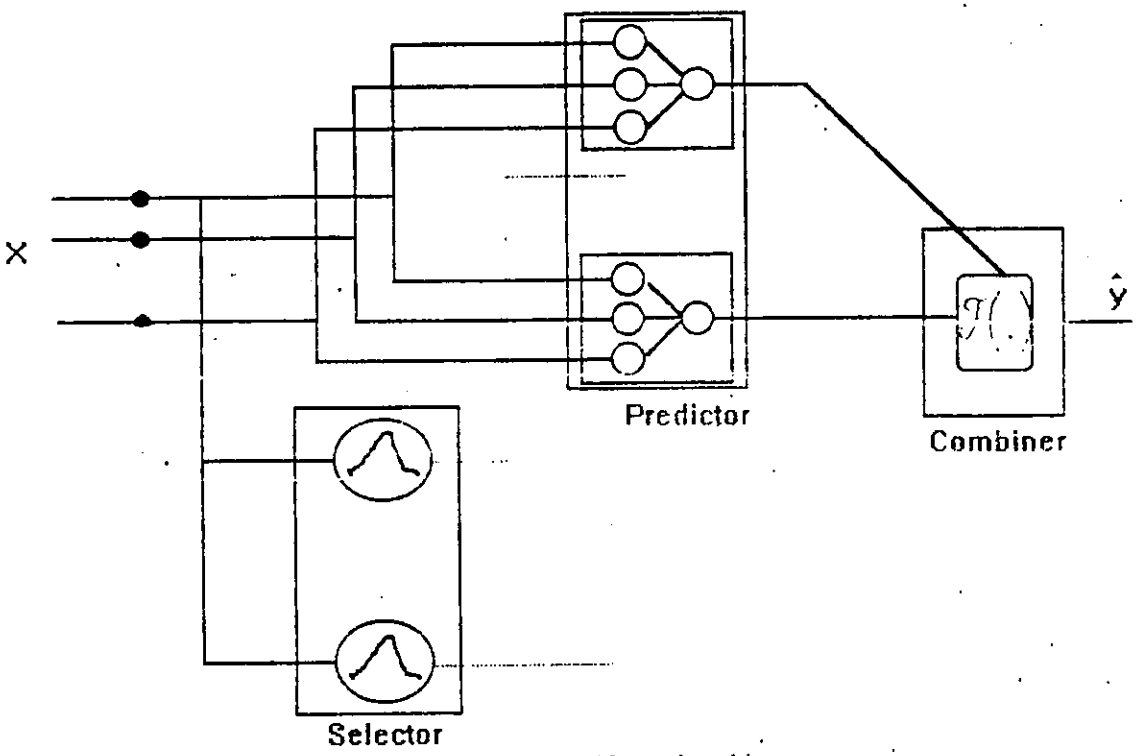


Figure 1 - SMP Network architecture

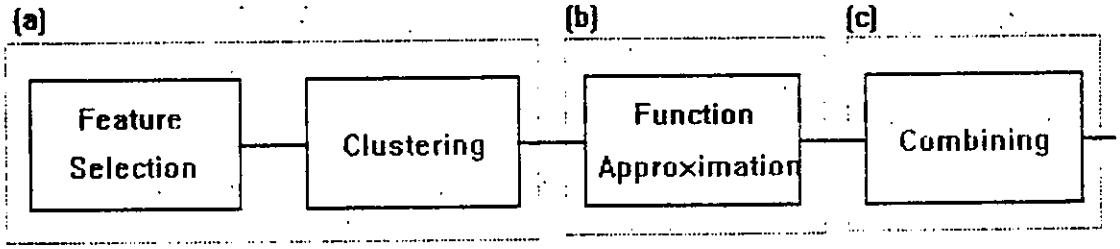


Figure 2 - SMP Processing Steps bloxk diagram. (a) pattern matching, (b) function approximation. (c) combining strategy.

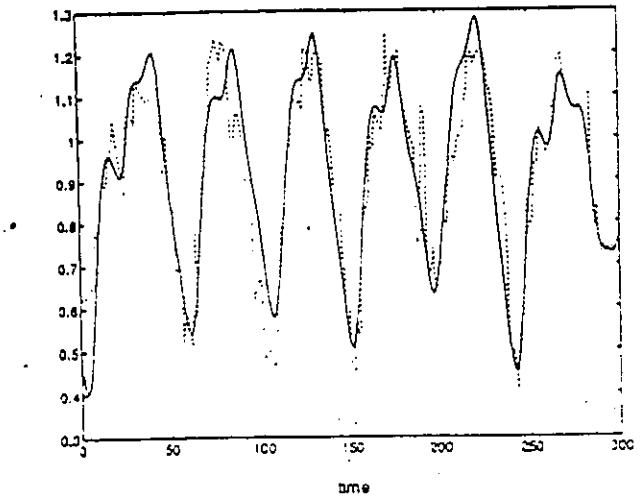
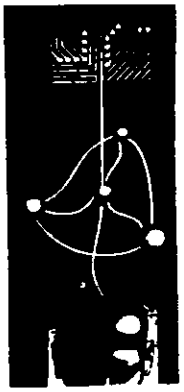


Figure 3 - WTA Committee Prediction on Mackey-Glass Time Series using MLP Networks with 7 units in the hidden layer and one output unit.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajuba
Itajuba, 24 a 27 de outubro de 1994

REDE NEURAL LÓGICA

A. L. C. Canella e L. P. Caloba

COPPE / EE / UFRJ
CP 68504 CEP 21945-970, Rio de Janeiro, Brasil
e.mail : caloba@coe.ufrj.br

RESUMO

Neste trabalho é apresentado um algoritimo construtivo para redes neurais com sinais de entrada e saída lógicos. A rede obtida é de duas camadas compostas por neurônios do tipo peceptron com função de ativação do tipo sinal. Todas as sinapses possuem valores inteiros. A rede obtida pode ser convertida num circuito digital com a mesma estrutura obtida pelo algoritimo de Quine-McCluskey.

1. INTRODUÇÃO

Sabe-se que qualquer função lógica pode ser implementada com portas lógicas numa estrutura digital de duas camadas, na qual utiliza-se apenas portas lógicas dos tipos E, OU, E negativo ou OU negativo[1]. Também sabe-se que com apenas um neurônio é possível implementar qualquer uma destas portas lógicas[2]. Logo, é coerente imaginar que qualquer função lógica possa ser implementada utilizando-se uma estrutura neural com apenas duas camadas[2].

O objetivo deste trabalho é apresentar um algoritimo construtivo de treinamento de rede neural com sinais de entrada e de saída lógicos. Este algoritimo monta uma rede neural de duas camadas de modo que cada neurônio da camada intermediária represente um mintermo da função lógica a ser implementada, realizando o neurônio da camada de saída a função lógica OU destes mintermos.

2. NEURÔNIO "E LÓGICO"

Neurônio E lógico é um dispositivo com entradas, saída e parâmetros com valores inteiros cuja saída fica ativa sse a entrada pertencer ao mintermo que o neurônio representa. O modelo de neurônio utilizado em tal finalidade é o do tipo perceptron com a função sinal como função de ativação[2].

Este dispositivo realiza a função lógica E através de uma sutileza matemática. A entrada é formada por n sinais lógicos representados pelos valores inteiros +1 e -1, correspondendo respectivamente aos valores lógicos verdadeiro e falso. As sinapses podem assumir os valores +1, -1 ou 0, com exceção da sinapse de polarização, que pode assumir outros valores inteiros. Cada parcela da soma de produtos que define a variável auxiliar u só pode valer -1, 0 e +1, com exceção da parcela correspondente a sinapse de polarização, que será analisada separadamente. Para que o neurônio j da primeira camada represente uma entrada lógica é suficiente fazer o vetor sinapse $\mathbf{w_j}$ igual ao vetor entrada $\mathbf{x}$. Desta forma, são positivas todas as parcelas da soma que define a variável auxiliar u correspondentes às sinapses ligadas aos sinais de entrada. Definindo-se apropriadamente o valor da sinapse de polarização w_0 é possível, utilizando-se estas características, realizar um E lógico entre o vetor de entrada $\mathbf{x}$ e o vetor sinapse $\mathbf{w_j}$.

O vetor sinapse $\mathbf{w_j}$ que representa os parâmetros do neurônio j da primeira camada é definido pela equação (1), na qual w_0 é a sinapse de polarização e w_i é a sinapse que liga o componente genérico x_i da entrada $\mathbf{x}$ ao neurônio j. O vetor $\mathbf{x}$ que representa a entrada é definido pela equação (2), -1 é o valor fixo da entrada de

polarização x_0 escolhido negativo por conveniência.

(1) $\mathbf{w}_j = \left[w_0\ w_1\ w_2 \cdots w_i \cdots w_{n-1}\ w_n\right]^t$

(2) $\mathbf{x} = \left[-1\ x_1\ x_2 \cdots x_i \cdots x_{n-1}\ x_n\right]^t$

A variável auxiliar u é definida pela equação (3) e a parte referente às parcelas correspondentes às sinapses que ligam os sinais de entrada ao neurônio, representada pela variável q, é definida pela equação (4). O valor desta variável é limitado pela relação definida na equação (5), na qual z é o número de sinapses nulas.

(3) $u = -w_0 + x_1w_1 + x_2w_2 + \cdots + x_iw_i + \cdots x_{n-1}w_{n-1}$

(4) $q = x_1w_1 + x_2w_2 + \cdots + x_iw_i + \cdots x_{n-1}w_{n-1} + x_nw_n$

(5) $q \le n - z$

Para que a entrada **x**, apresentada ao neurônio j da primeira camada, pertença ao mintermo por ele representado, todas as parcelas de q devem ser positivas ou nulas. Por outro lado, para que a entrada não pertença a este mintermo, pelo menos uma parcela de q deve ser negativa. As equações (6) e (7) definem matematicamente os dois casos, sendo k o número de parcelas negativas de q.

(6) $q = n - z$

(7) $q = n - z - 2k$

A sinapse de polarização w_0 é definida de tal forma que u seja positivo quando **x** pertencer ao mintermo representado pelo neurônio j e negativo quando não pertencer. A equação que relaciona a variável auxiliar u, a variável q e a sinapse de polarização w_0 é obtida das equações (3) e (4) e é representada pela equação (8).

(8) $u = q - w_0$

Aplicando a equação (6) na equação (8) obtem-se a relação dada pela equação (9), na qual é definida a faixa de valores que w_0 pode assumir para tornar a saída do neurônio j positiva quando **x** pertencer ao mintermo por ele representado. Por outro lado, aplicando a equação (7) na equação (8) obtem-se a relação dada pela equação (10) na qual é definida a faixa de valores que w_0 pode assumir para tornar a saída do neurônio j negativa quando **x** não pertencer ao mintermo por ele representado.

(9) $w_0 < n - z$

(10) $w_0 > n - z - 2k$

Unindo as relações dadas pelas equações (9) e (10) obtem-se a relação dada pela equação (11), na qual é definida a faixa de valores possíveis de w_0 para satisfazer ambas relações. No caso limite, onde k é igual a 1, o único valor inteiro que satisfaz esta relação é indicado pela equação (12).

(11) $n - z - 2k < w_0 < n - z$

(12) $w_0 = n - (z + 1)$

Utilizando-se este valor inteiro para w_0, dependente apenas da ordem da entrada e da ordem do neurônio (número de sinapses nulas), iguais respectivamente a n e z, ao aplicarmos a função sinal sobre a variável auxiliar u, transformamos um simples neurônio do tipo perceptron em um dispositivo que determina a pertinência da entrada a um mintermo.

3. SUPERFÍCIE SEPARADORA

Substituindo na equação (3) o valor inteiro de w_0 definido pela equação (12) e igualando a variável auxiliar u à zero, temos a definição da superfície separadora dada pela equação (13) [2].

(13) $x_1w_1 + x_2w_2 + \cdots + x_iw_i + \cdots x_{n-1}w_{n-1} + x_nw_n + z - n + 1 = 0$

Os pontos de interseção do hiperplano separador com os eixos cartesianos são determinados pela equação (14), pela qual se verifica não haver interseção com os eixos relativos às sinapses nulas e, caso contrário, a interseção ocorre em pontos inteiros com módulo definido pela equação (15).

(14) $x_i = \dfrac{z + 1 - n}{w_i}, \quad w_i \neq 0 \ \text{ e } \ x_j = 0 \ \ \forall \ j \neq i$

(15) $|x_i| = z + 1 - n, \quad w_i \neq 0$

Um mintermo formado por z sinapses nulas é representado por um sub-hipercubo de ordem z contido no hipercubo de ordem n. O número de arestas de ligação *nal* que conectam este sub-hipercubo de ordem z aos demais vértices do

hipercubo de ordem n é o número de arestas cortadas pelo hiperplano separador. Como cada vértice de um hipercubo de ordem n é ligado a n outros vértices e os que formam o sub-hipercubo de ordem z já estão ligados a z vértices dele próprio, conclui-se que cada um dos 2^z vértices do mesmo é ligado a $(n\text{-}z)$ vértices restantes do hipercubo de ordem n, definindo-se desta forma a equação (16) que determina o número *nal*. A forma genérica das entradas representadas por um mintermo realizado por um neurônio E lógico é fornecida pela equação (17), sendo e_i igual a w_i se w_i for diferente de zero e podendo ser ± 1 se w_i for igual a zero.

(16) $$nal = (n-z)2^z$$

(17) $$\mathbf{e} = \left[-1\ e_1\ e_2 \cdots e_i \cdots e_{n-1}\ e_n\right]$$

Como há z sinapses iguais a zero, existem z entradas "don't care" em **e**, determinando-se desta forma 2^z vetores que satisfazem a equação (17). Aplicando-se o vetor genérico **e** em u obtem-se a equação (18).

(18)
$$u = e_1 w_1 + e_2 w_2 + \cdots + e_i w_i + \cdots + e_{n-1} w_{n-1} + e_n w_n + z - n + 1 = +1$$

Como existem $(n\text{-}z)$ sinapses não nulas em (18), igualar a zero apenas uma das $(n\text{-}z)$ entradas correspondentes a estas sinapses, significa zerar a variável auxiliar u com uma entrada correspondente à um ponto localizado no centro de uma aresta. Como existem $(n\text{-}z)$ posibilidades e para cada uma há 2^z formas diferentes, devido às entradas "don't care", pode-se concluir que existem no total $(n\text{-}z)2^z$ pontos localizados em centros de arestas que anulam u. Ou seja, o hiperplano cruza $(n\text{-}z)2^z$ centros de arestas. Como o mesmo só cruza $(n\text{-}z)2^z$ arestas do hipercubo, conclui-se que o mesmo só o cruza nos centros das arestas.

4. NEURÔNIO "OU LÓGICO"

A implementação do neurônio OU lógico é bastante simples e deriva diretamente do neurônio E lógico. Imagina-se um hipercubo com n^2 vértices, cria-se um neurônio E lógico que separe apenas o vértice correspondente a todas as entradas negativas, ou seja, todas as entradas com nível lógico falso. Este neurônio E lógico fornecerá saída positiva quando todas as entradas tiverem nível lógico falso e saída negativa quando pelo menos uma entrada possuir nível lógico verdadeiro. Deste modo, é implementado um OU lógico negativo. Invertendo o sinal de saída deste neurônio E lógico, o que representa multiplicar por -1 todos os valores das sinapses, obtem-se um neurônio OU lógico, em que w_i=+1 $\forall$ i e w_0=1-n.

5. TREINAMENTO

O treinamento é um processo iterativo onde apenas as entradas com saídas verdadeiras são consideradas e usadas no treinamento, e apenas uma única vez. Para cada entrada, inicialmente é formado um hiperplano que a separa dos demais vértices, e é iniciado um processo iterativo convergente que determinará todos os hiperplanos que separam o maior número de vértices possivel que esta entrada pode formar com as já treinadas. Este hiperplano inicial é representado por um neurônio que possui a seguinte regra de formação:

(19) $$\mathbf{WB} = [w_0\ w_1\ w_2 \ldots w_i \ldots w_{n-1}\ w_n]$$

fazendo-se,

$w_0 = n\text{-}1$

$w_i = x_i$

onde,

x_i : é a componente i do vetor de entrada;

Os neurônios são divididos em 3 grupos: o grupo A possui todos os neurônios existentes antes da nova entrada ser treinada; o grupo B é inicialmente constituído pelo neurônio WB, equação (19), criado pela nova entrada; o grupo C é formado pelos neurônios gerados quando se compara um neurônio do grupo B com um do grupo A.

5.1 Combinação de hiperplanos.

A seguir é descrito o método utilizado para comparar e gerar os novos hiperplanos.

Sejam **WA**, **WB** e **WC**, respectivamente, um neurônio do grupo A, um do grupo B e o a ser gerado no grupo C. Cujos vetores representativos são os seguintes:

$$\mathbf{WA} = [\ wa_0\ wa_1\ wa_2 \ldots wa_i \ldots wa_{n-1}\ wa_n\ ]$$
$$\mathbf{WB} = [\ wb_0\ wb_1\ wb_2 \ldots wb_i \ldots wb_{n-1}\ wb_n\ ]$$
$$\mathbf{WC} = [\ wc_0\ wc_1\ wc_2 \ldots wc_i \ldots wc_{n-1}\ wc_n\ ]$$

O algorítimo de comparação e associação é o seguinte:

0) $z = 0$
1) se $wb_0 < wa_0$, então:
sai;
2) repetir sequencialmente para i igual a 1 até n
2.a) se $wa_i = 0$ e $wb_i \neq 0$, então:
sai;
2.b) $wc_i = wa_i + wb_i$
se $wc_i > 0$, então $wc_i = +1$;
se $wc_i < 0$, então $wc_i = -1$;
se $wc_i = 0$, então $z = z + 1$;
2.c) se $z > 1$, então:
sai;
3) se $z \neq 1$, então:
sai;
4) $wc_0 = wa_0 - 1$
5) se $wa_0 = wb_0$, então:
elimina-se o neurônio **WA**;
6) incluir o novo neurônio **WC**.

5.2 Algorítimo de Treinamento

Cada neurônio do grupo B é comparado com todos os neurônios do grupo A e os neurônios gerados nesta comparação formam o grupo C. Após todos os neurônios do grupo B serem comparados com os do grupo A, os neurônios restantes do grupo B são unidos aos restantes do grupo A. E os gerados no grupo C passam a formar o grupo B. Este processo iterativo continua até que não haja a formação de novos neurônios no grupo C.

O algoritimo consiste no seguinte:

0) verificar se ainda há entradas a serem treinadas: se houver então continuar, caso contrário, ir para o passo 10;
1) considerar a próxima entrada;
2) se a saída correspondente for falsa, então retornar para o passo 0, caso contrário, continuar;
3) gerar um neurônio que forme um hiperplano que separe o vértice correspondente a esta entrada das demais;
4) iniciar o grupo B com este neurônio;
5) verificar se há neurônios no grupo B: se houver continuar, caso contrário, retornar ao passo 0;
6) comparar os neurônios do grupo B com os do grupo A e formar novos neurônios, se possível, no grupo C;
7) formar um novo grupo A com os neurônios restantes dos grupos A e B;
8) formar um novo grupo B com os novos neurônios gerados no grupo C;
9) retornar ao passo 5;
10) iniciar a etapa de secagem;

Durante o treinamento é necessário realizar simultaneamente as seguintes secagens (eliminação de neurônios supérfluos):

a) após um neurônio do grupo B ter sido comparado com todos os neurônios do grupo A, ele deve ser eliminado caso tenha gerado algum novo neurônio no grupo C;
b) ao se comparar um neurônio do grupo B com um do grupo A e houver a geração de um novo neurônio no grupo C, o neurônio do grupo A deve ser eliminado caso a sua ordem seja igual a do neurônio do grupo B;
c) só deve ser gerado um novo neurônio no grupo C caso o mesmo ainda não tenha sido gerado neste grupo;

Com o objetivo de acelerar o processo, os neurônios do grupo B só devem ser comparados com os do grupo A que tiverem ordem igual ou superior.

5.3 Secagem

O método consiste na formação de um grupo I a partir dos elementos do grupo II que inicialmente possui todos os neurônios obtidos na etapa de treinamento, isto é, os neurônios do grupo A. O objetivo é formar o grupo I com apenas os elementos indispensáveis do grupo II para a representação da função lógica a ser implementada.

O algoritimo é bastante simples e eficiente. Consiste no transporte para o grupo I de todos os neurônios do grupo II cujos hiperplanos separem pelo menos um vértice não separado pelos outros hiperplanos. Quando não houver nenhum neurônio com esta caracteristica, elimina-se o neurônio de menor significância do grupo II. Se persistir em não haver nenhum neurônio do grupo II habilitado para ser transportado para o grupo I, elimina-se outro neurônio de menor significância do grupo II. Quando houver algum neurônio do grupo II que possa ser transportado para o grupo I, realiza-se a transferência. Este processo segue sucessivamente até não haver mais neurônios no grupo II.

Este algoritimo é de facil implementação. Cada iteração consiste basicamente de quatro

etapas sequênciais. São realizadas quantas iterações forem necessárias para eliminar ou transferir para o grupo I todos os neurônios do grupo II.

A primeira etapa consiste em determinar para cada neurônio do grupo II três parâmetros. O primeiro, denominado de P1, é o número de vértices separados pelo hiperplano correspondente ao neurônio que é igual a 2^{-z}, onde z é a ordem ou o número de sinapses nulas. O segundo, denominado de P2, é o número de vértices, entre os separados pelo hiperplano correspondente ao neurônio em questão, não separados por nenhum dos hiperplanos correspondentes aos neurônios do grupo I. O terceiro, denominado de P3, é o número de vértices, entre os separados pelo hiperplano correspondente ao neurônio em questão, não separados pelos hiperplanos correspondentes aos neurônios do grupo I e aos demais neurônios do grupo II.

O algoritimo que determina os valores dos parâmetros P1, P2 e P3 é bastante simples e direto. Sabendo-se o número de sinapses nulas e quais são elas, a determinação de quantas e quais são as entradas englobadas pelo neurônio em questão é direta. Deste modo, a determinação do valor do parâmetro P1 é direta e é igual a 2^{-z}. Para se determinar os valores dos parâmetros P2 e P3 deve-se considerar duas redes neurais. A rede I é formada pelos neurônios do grupo I e a rede II é formada pelos neurônios que formam o grupo II com exceção do neurônio em questão. Realiza-se um loop com P1 iterações, onde é apresentada a ambas redes as P1 entradas englobadas pelo neurônio em questão. O valor do parâmetro P2 é determinado pelo número de entradas que não ativam a rede I, ou seja, pelo número de entradas ainda não englobadas por algum dos neurônios do grupo I. O valor do parâmetro P3 é determinado pelo número de entradas que não ativam simultaneamente as redes I e II, ou seja, pelo número de entradas não englobadas pelos neurônios dos grupos I e II.

A segunda etapa consiste simplesmente na eliminação de todos os neurônios do grupo II que possuam o parâmetro P2 nulo, pois, estes não possuem nenhuma informação nova a ser acrescentada ao grupo I.

A terceira etapa é o transporte para o grupo I de todos os neurônios artificias do grupo II com o parâmetro P3 não nulo, pois, estes possuem alguma informação nova a ser acrescentada no grupo I não contida em nenhum outro neurônio do grupo II.

A quarta etapa consiste na eliminação do neurônio de menor significância do grupo II, caso não tenha ocorrido nenhuma transferência para o grupo I. Ou seja, elimina-se um dos neurônios que obtiverem o menor parâmetro P1 entre os neurônios do grupo II com o menor parâmetro P2.

Este algoritimo é repetido sucessivamente até que não haja mais neurônios no grupo II. Como em cada iteração pelo menos um neurônio do grupo II é eliminado ou transferido para o grupo I, este é um processo iterativo convergente, pois o número de neurônios do grupo II é finito.

6. ENTRADAS "DON'T CARE"

Será realizada a construção de duas redes neurais simultaneas. Na rede principal serão treinadas as entradas com saída verdadeira e entradas "don't care" com saída considerada como verdadeira. A rede auxiliar será treinada apenas com entradas "don't care" com saída considerada como verdadeira. Desta forma, teremos na rede principal a representação neural de todos os maiores mintermos possíveis de serem formados com entradas cuja saída seja verdadeira e com entradas "don't care" com saída considerada como verdadeira. Também, estará representado na rede principal todos os maiores mintermos possíveis de serem formados apenas com entradas "don't care" com saída considerada como verdadeira. Na rede auxiliar estará representado todos os maiores mintermos que as entradas "don't care" com saída considerada com verdadeira podem formar.

A etapa de secagem, além de ser responsável por eliminar os mintermos dependentes, será também responsável pela eliminação dos termos formados apenas por entradas "don't care" com saída considerada como verdadeira. Desta forma, restará na rede principal apenas os maiores mintermos que possuam alguma entrada com saída definida. Para que isto ocorra, a etapa de secagem deve ser alterada. Esta alteração ocorre na primeira etapa do algoritimo iterativo responsável pela seleção e eliminação dos elementos do grupo II. Acrescenta-se um grupo III formado pelos neurônios da rede auxiliar. O cálculo do parâmetro P1 é o mesmo, mas, os cálculos dos

parâmetros P2 e P3 devem ser alterados. O parâmetro P2 passa a ser o número de vértices, entre os separados pelo hiperplano correspondente ao neurônio do grupo II em questão, não separados pelos hiperplanos correspondentes aos neurônios dos grupos I e III. O parâmetro P3 passa a ser o número de vértices, entre os separados pelo hiperplano correspondente ao neurônio do grupo II em questão, não separados pelos hiperplanos correspondentes aos neurônios dos grupos I e III e pelos demais neurônios do grupo II.

7. EXEMPLOS

São apresentados dois exemplos de [1], onde as funções lógicas são definidas por um somatório de mintermos: m indica entradas com saídas lógicas verdadeiras e d indica entradas "don't care". Os resultados obtidos são apresentados nas tabelas abaixo.

$F(A,B,C,D,E) = \Sigma$ m(0,6,8,10,12,14,17,19,20, 22,25, 27,28,30)

w0	w1	w2	w3	w4	w5	mint.
3	-1	-1	-1	0	-1	$\overline{A}\,\overline{C}\,\overline{D}\,\overline{E}$
2	-1	0	0	1	-1	$\overline{A}B\overline{E}$
2	1	0	-1	0	1	$A\overline{C}E$
2	-1	1	1	0	0	$CD\overline{E}$
2	-1	0	1	0	1	$AC\overline{E}$

$F(A,B,C,D,E) = \Sigma$ m(1,4,7,14,17,20,21,22,23) + d(0,3,6,19,30)

w0	w1	w2	w3	w4	w5	mint.
2	-1	1	1	0	0	$CD\overline{E}$
2	1	0	-1	-1	0	$\overline{B}\,\overline{C}E$
2	-1	0	1	-1	0	$\overline{B}C\overline{E}$
2	0	1	1	-1	0	$\overline{B}CD$
2	0	0	1	-1	1	$A\overline{B}C$

8. CONCLUSÕES

O resultado final obtido pelo algoritmo apresentado é o mesmo que seria obtido pelo algoritmo de Quine-McCluskey caso a rede obtida fosse implementada com portas lógicas. Existem outros algoritimos construtivos que podem obter estruturas de redes neurais mais compactas. Entretanto, com relação a implementação, não são tão simples e confiáveis quanto aos circuitos digitais[3]. A estrutura neural proposta é simples de ser implementada e tão confiável e compacta quanto um circuito digital obtido pelo uso do algoritimo de Quine-McCluskey.

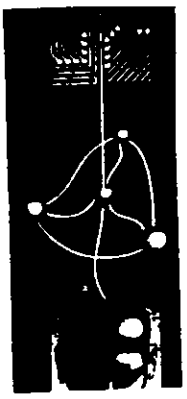
A diferença básica entre os dois algoritmos é que enquanto o de Quine-McCluskey opera sobre toda a massa de dados simultaneamente, necessitando a utilização de grandes tabelas, o algoritmo apresentado neste trabalho processa cada entrada individualmente e um só vez. Este processamento dos dados de entrada um a um é usualmente um requisito dos sistemas neurais. Assim, pode-se supor que o algoritimo de treinamento proposto se aproxima muito mais de um modelo neural natural que um algoritimo de treinamento diretamente originado do algoritimo de Quine-McCluskey, alem da necessitar de muito menos memória.

9. REFERÊNCIAS

[1] F. F. Hill e G. R. Peterson, "Introduction to Switching Theory & Logical Design", John Willey & Sons, 1981.

[2] J. Hertz, A. Krogh e R. G. Palmer, "Introduction to the Theory of Neural Computation", Addison Wesley, 1990.

[3] H. M. A. Andree, G. T. Barkema, W. Lourens e A. Taal, "A Comparison Study of Binary Feedforward Neural Networks and Digital Circuits", Neural Networks, Vol. 6, pp. 785-790, 1993.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajuba
Itajuba, 24 a 27 de outubro de 1994

Redes RAM Radial

ANNE MAGÁLY DE PAULA CANUTO[1]
EDSON COSTA DE BARROS CARVALHO FILHO[1]

[1]UFPE-Universidade Federal de Pernambuco
Departamento de Informática
Cx 7851, 50.732-970, Recife, PE, Brasil
ampc@di.ufpe.br e ecdbcf@di.ufpe.br

Sumário. O modelo de redes neurais conhecido como RAM-Based é capaz de computar qualquer função booleana com um dado número de entradas. Neste artigo, é proposto e investigado o modelo RAM Radial, que é uma generalização do RAM original, diferenciando no estado de uso. Neste estado, quando uma entrada é apresentada ao neurônio, não apenas o conteúdo endereçado, conhecido como ponto central, é acessado, mas sim toda uma região radial. A análise do desempenho da rede mostra que o RAM Radial obteve melhores resultados, sendo sempre *mais confiável* que o RAM original. As implicações destes resultados vão além da área de redes neurais, aplicando-se na área de reconhecimento de padrões com uma generalização da técnica de n-tupla.

1 Introdução

Os modelos de redes neurais são inspiradas no sistema nervoso biológico e são compostos de elementos(neurônios) que funcionalmente são análogos a modelos simplificados de neuronios biológicos. Existe uma classe de redes neurais, conhecida como RAM devido aos seus neurônios terem uma estrutura similar a de uma memória de acesso aleatório, que é capaz de computar qualquer função booleana com um dado número de entradas [3, 5].

Neste artigo, será apresentado o modelo de rede RAM Radial, que é uma generalização do RAM original diferindo na forma de acesso aos conteúdos. O modelo proposto não acessa apenas um endereço para calcular a saída, mas sim toda uma região radial deste endereço. A proposta do modelo RAM Radial emerge de estudos no modelo Goal Seeking Neuron (GSN) [1, 2]

Este trabalho está dividido em quatro partes: a primeira parte será realizada uma breve descrição do modelo RAM radial; a segunda parte será mostrada a arquitetura da rede; a terceira parte será mostrada uma análise do desempenho dos experimentos realizados com a rede; e a quarta e última parte serão feitas as considerações finais sobre o modelo.

2 Neurônio RAM Radial

O modelo RAM [3] pode ser visto como uma técnica de implementação de um processo de reconhecimento primeiramente descrito por Bledsoe e Browning em 1959 [9]. Este processo é conhecido como o método da n-tupla. O termo **n-tupla** vem do fato que cada unidade (neurônio) recebe n entradas como uma tupla.

O neuronio RAM radial se assemelha muito com o neurônio RAM no fato que seus neurônios recebem valores binários e são capazes de computar qualquer função booleana com um dado número de entradas.

A diferença entre os modelos RAM Radial, que está sendo analisado neste trabalho, e o RAM está no estado de uso. Neste estado, quando uma entrada é apresentada ao neuronio, não apenas o conteúdo endereçado, conhecido como ponto central, é acessado, mas sim toda uma região radial(região de acesso). Esta região é composta pelos conteúdos cujo endereço está até uma determinada distância do ponto central, conhecidos como conjunto endereçáveis.

Se os pesos forem rotulados como 2^k onde $0 \leq k \geq c_i - 1$, então o ponto central, a_{ik}^c, acessado por uma entrada é definido pela equação 1.

$$a_{ik}^c = \sum_{j=1}^{j=c_i} x_{ij} z_{ij} \qquad (1)$$

onde x_{ij} é o valor da entrada do j-ésimo terminal de entrada e z_{ij} é o peso do j-ésimo terminal.

O cálculo de todos os endereços do conjunto endereçável pode ser visto na equação 2.

$$A_{im} = \{a_{im} = \sum_{j=1}^{j=c_i} x_{ij} z_{ij}, \forall a_{im} \mid D(a_{im}, a_{ik}^c) \leq R\} \qquad (2)$$

onde :

- R é o valor da distancia máxima;
- c_i é a conectividade do neuronio;
- a_{ik}^c é o endereço do ponto central da região;
- a_{im} são os endereços dos conteúdos da região;

Este conjunto endereçável pode ser genérico, como todos os pontos de uma esfera ou cubo, ou analisando a distribuição de probabilidade do conjunto de

dados. Uma vez que os endereços estão distribuidos ao longo de uma distância escolhida para a formação da região, utiliza-se uma função de ponderação para diferenciar a contribuição dos conteúdos acessados na saída do neuronio. Vários tipos de funções podem ser utilizadas, tais como: a função linear, a função gaussiana, a função hiperbólica, etc, que encontram presupostos biológicos e são utilizadas nos modelos de Linsker e Kohonen [7, 8].

A função de ativação é computada em cima do conjunto endereçável com seus respectivos pesos. Uma forma genérica que pode ser útil no cálculo da saída do neurônio radial é dada na equação 3.

$$o_i = F(s_i) \tag{3}$$

onde :

$$s_i = \sum_{\forall a_{im} \in A_i} G(C[a_{im}], D(a_{im}, a^c_{ik})$$

e :

o_i é a saída do neurônio i;
$C[a_{im}]$ é o conteúdo do endereço a_{im};
F é a função de ativação;
G é a função de ponderação;

A função G pondera valores com o objetivo de valorizar alguns conteúdos (C[a_{im}]) da região, dependendo da distância (D) do ponto central.

3 Uma Arquitetura para Reconhecimento

Uma arquitetura que pode ser utilizada com o modelo RAM Radial é uma rede de discriminadores. Esta rede é formada por apenas uma camada de neurônios. A escolha desta rede é porque ela se assemelha com o método da n-tupla. A utilização do modelo RAM Radial nesta arquitetura representa uma modificação na técnica de n-tupla original.

Na arquitetura de discriminadores de uma camda, o processo de aprendizagem é supervisionado, associando cada discriminador com uma classe de padrão [6]. A figura 1 ilustra a arquitetura utilizada na rede.

Neste tipo de arquitetura, cada discriminador possui um número fixo de neurônios e é associado com uma classe para aprendê-la, e os neurônios possuem uma conectividade fixa [5, 6].

3.1 A Fase de Aprendizagem

No início, todos os neurônios de todos os discriminadores são zerados. Quando a rede entra na fase de aprendizagem, ela associa um discriminador com uma classe de padrão. Nesta fase, associa-se todos os padrões de treinamento de uma classe a um discriminador e ensina a este discriminador aquela classe, este ensino é feito colocando 1 nos conteúdos acessados pelas entradas em todos os neurônios do discriminador selecionado, ou seja, apenas o discriminador selecionado aprende a classe.

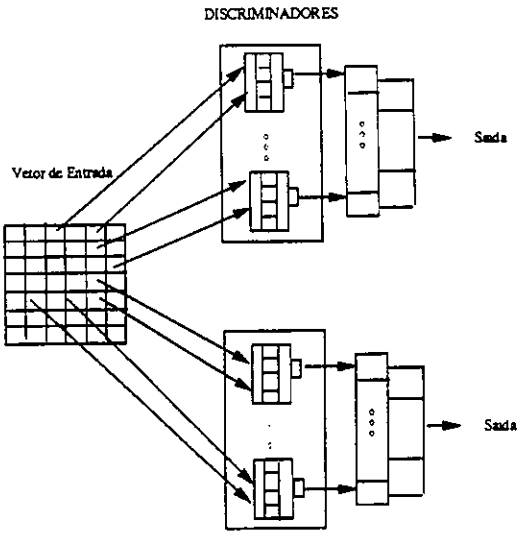


Figura 1: *A Arquitetura de discriminadores com RAM Radial*

3.2 A fase de teste

Quando um padrão de teste é apresentado a rede, todos os neurônios de todos os discriminadores são ativados e produzem uma saída, esta saída indica o quão similar a região de acesso é da entrada. A saída do discriminador é calculada somando todas as saídas dos seus neurônios. O discriminador que possuir a maior saída é aquele que melhor representa o padrão de entrada.

4 Análise do desempenho

Para investigar as prorpiedades fundamentais e habilidades da rede RAM Radial, um conjunto de dados foi selecionado para ser utilizado nos experimentos . O conjunto de dados escolhido é formado por caracteres digitalizados alfanuméricos de 0 a 9 podendo ser extendido para todas as letras e caracteres manuscritos. Cada classe contém 300 padrões digitalizados em 384 pontos brancos e pretos numa matriz de 24 X 16.

Nestes experimentos, a função de ativação utilizada foi a função sigmoide e a função de ponderação foi a gaussiana. A função de ponderação valoriza os pontos que estão próximos do ponto central. As funções podem ser vista nas equações 4 e 5 respectivamente.

$$F(s_i) = \frac{1}{1 + exp(-s_i/\theta)} \tag{4}$$

$$G(x,y) = \begin{cases} exp(\frac{-y^2}{2*\delta^2}) & \text{se } C_{ij} \text{ for 1} \\ -exp(\frac{-y^2}{2*\delta^2}) & \text{se } C_{ij} \text{ for 0} \end{cases} \tag{5}$$

onde:

θ estabelece a largura da sigmoide

δ é o sigma da gaussiana e estabelece a largura da gaussiana.

O desempenho da rede será analisado em termos de respostas corretas, erradas e rejeitadas, similaridade e confiabilidade. A similaridade é calculada através do valor do desempenho de respostas corretas do discriminador vencedor de todos os padrões. A confiabilidade é calculada pela diferença de desempenho do discriminador vencedor e o segundo maior discriminador.

As próximas seções demonstrarão a análise comparativa do desempenho das redes com a variação do conjunto de treinamento, o sigma da gaussiana, o número de neuronios por discriminador, a conectividade do neurônio e o peso dos conteúdos radiais.

O objetivo desta análise é verificar se ocorreu uma melhora no nível de reconhecimento utilizando o modelo RAM Radial que representa uma contribuição no modelo RAM e n-tupla.

4.1 Número de neurônios por discriminadores

Com o objetivo de analisar o desempenho da rede variando a quantidade de neuronios por discriminador, foram implementadas várias redes, todas elas posuiam 10 discriminadores onde cada discriminador possuia 10, 50, 100, 150 e 200 neurônios por discriminador com conectividade 4. Estas redes foram treinadas com 64 padrões por classe e recuperadas com 100 padrões por classe. Os padrões foram apresentados na ordem de 0 a 9, apresentando todos os padrões da classe 0, depois todos da classe 1 e vai até todos da classe 9. O sigma da gaussiana utilizada para cáculo dos pesos foi 0.5.

A figura 2 ilustra o desempenho de reconhecimento em termos de respostas corretas, erradas e rejeitadas. A simbologia RB e RBR utilizada a partir desta figura significa RAM e RAM Radial respectivamente.

Os resultados obtidos mostram que com o número de neurônios igual a 10, a diferença de desempenho das duas redes praticamente não existe, isto é devido a pequena área coberta, 40/384. Com o número de neurônios igual a 50 esta diferença aumenta devido a um melhor desempenho de respostas corretas da RAM Radial. No entanto, com número de neurônios maior que 50 esta diferença vai diminuindo até ficar constante.

Agora, será visto o desempenho da rede em relação a similaridade e a confiabilidade. A figura 3 ilustra o desempenho das redes.

Os resultados obtidos nesta análise mostram que as similaridades da RAM Radial está quase sempre abaixo da RAM, apenas na similaridade de rejeição que a rede é maior com 10, 50, abaixando logo em seguida. Isto ocorre pois para uma neuronio responder

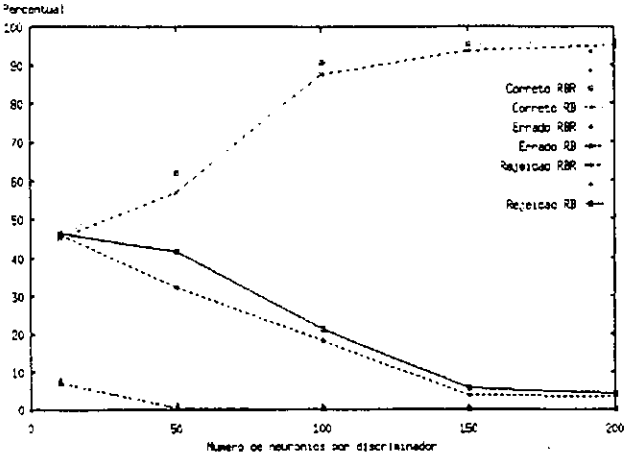


Figura 2: *O Desempenho de respostas corretas erradas e rejeitadas com variação do númer de neurônios*

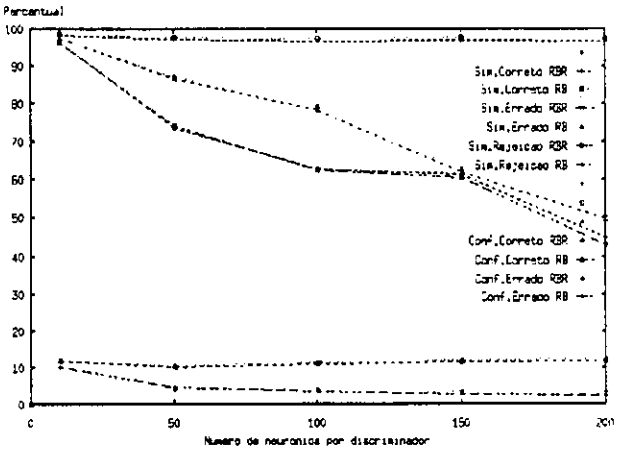


Figura 3: *O Desempenho de similaridade e Confiabilidade com variação do número de neurônios*

1, todos os endereços da região devem conter 1 e isso dificulta a resposta 1 dos neurônios, geralmente a saída fica entre 0.8 e 0.9, já o RAM normal responde 1 se o seu conteúdo possuir 1. Agora, a confiabilidade do RAM radial é sempre melhor que o RAM-based, esta diferença se torna maior com o aumento do número de neuronios.

A primeira vista, se pode concluir que o melhor desempenho da RAM Radial é com o número de neuronios igual a 50, no entanto, a área coberta desta rede é pequena, 200/384, e apenas pouco mais da metade do padrão está sendo pesquisado pelos neurônios. Diante disso, pode-se notar que, em relação ao número de neurônios, o melhor desempenho da RAM Radial é com um número de neurônios igual a 100.

4.2 O Conjunto de Treinamento

Para verificar o desempenho da rede com diferentes tamanhos conjuntos de treinamento, a rede descrita

anteriormente foi treinada com 1, 2, 4, 8, 16, 32 e 64 padrões por classe. Ao verificar os resultados, figura 4, pode-se ver que com o aumento do conjunto de treinamento, a diferença de desempenho, favorecendo o RAM Radial, vai aumentando. O aumento dessa diferença ocorre devido ao número crescente de 1's nos conteúdos dos endereços da região de acesso, dos discriminadores treinados com aquele padrão, com o aumento do conjunto de treinamento. Ao verificar o desempenho de respostas erradas e rejeitadas, pode-se notar que está havendo uma troca, as respostas que eram erradas, com o aumento do conjunto de treinamento, estão sendo respostas rejeitadas.

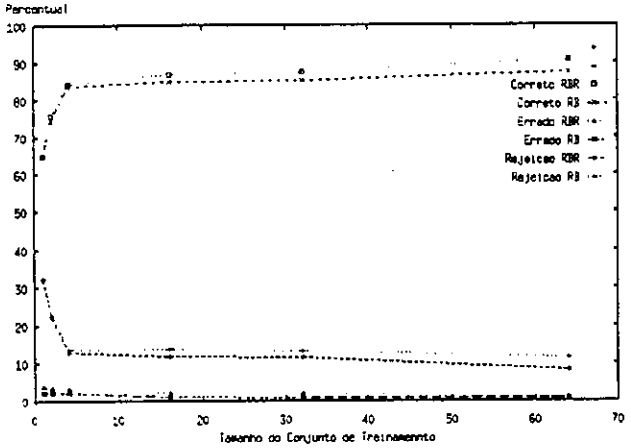


Figura 4: *O desempenho de respostas corretas erradas e rejeitadas com variação do conjunto de treinamento*

A figura 5 ilustra o desempenho da rede em relação a similaridade e confiabilidade. Os resultados mostram que com o aumento do conjunto de treinamento a diferença de desempenho, no início o RAM é melhor, vai diminuindo até praticamente não existir diferença. Agora, ao verificar a confiabilidade pode-se notar que com os conjuntos de treinamento pequenos, a RAM Radial é menos confiável que o RAM, mas com o aumento do conjunto de treinamento o RAM Radial passa a ser mais confiável.

Após toda essa análise, pode-se concluir que a rede que obteve os melhores resultados foi a rede treinada com 64 padrãoes por classe pois *tem o maior desempenho de respostas corretas e é a mais confiável.*

4.3 A Conectividade

Para analisar o desempenho da rede com conectividades diferentes, foram treinadas várias redes, semelhantes à descrita anteriormente, com conectividade 2, 4, 6 e 8, com área coberta de 200/384, 400/384, 600/384 e 800/384 respectivamente.

Os resultados, figura 6, mostram que no início o RAM Radial tinha melhor percentagem de respostas

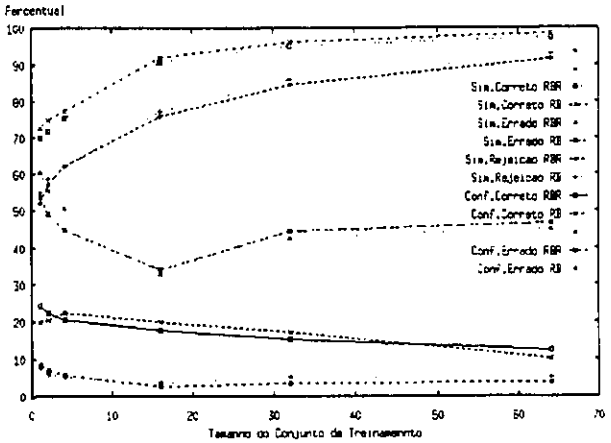


Figura 5: *O Desempenho de similaridade e Confiabilidade com variação do conjunto de treinamento*

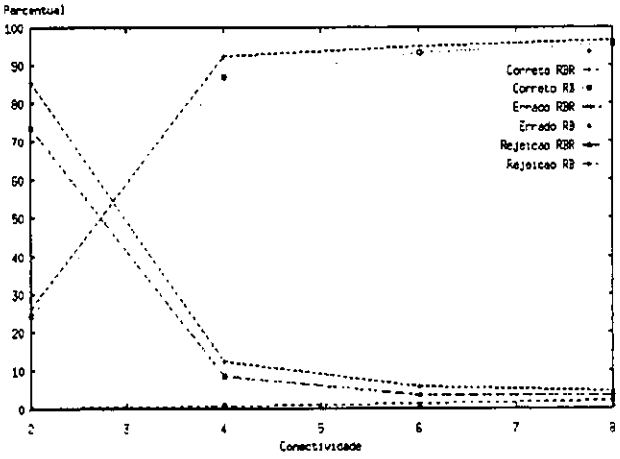


Figura 6: *O Desempenho de respostas corretas erradas e rejeitadas com variação da conectividade*

corretas e com o aumento da conectividade o RAM vai se aproximando do RAM Radial até praticamente não existir dieferença com conectividade 8. Esta diminuição ocorre pois com o aumento da conectividade há um aumento exponencial nos endereços dos neuronios e como o conjunto de treinamento e a quantidade de endereços acessados continuam fixos então ocorre um esparsamento de endereços acessados dentro do neurônio, é o mesmo processo que ocorre com conjuntos de treinamento pequenos.

A figura 7 ilustra o comportamento da rede em relação a similaridade e confiabilidade. Os resultados obtidos mostram que todas as similaridades e confiabiliades não tem diferença entre as duas redes com conectividade baixa e essas diferenças são diretamente proporcionais ao aumento da conectividade, sendo o RAM-Based Radial mais confiável e menos similar.

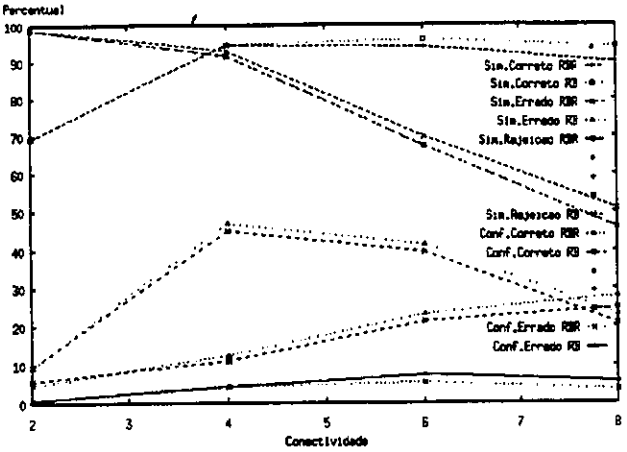


Figura 7: *O Desempenho de similaridade e Confiabilidade com variação da conectividade*

Com esses resultados, pode-se concluir que apesar de ser menos confiável, a rede com conectividade 4 é a melhor configuração pois as similaridades estão mais próximas do RAM e com uma conectividade maior há um maior consumo de tempo e memória.

4.4 O Peso dos conteúdos radiais

A variação dos pesos dos conteúdos radiais é calculada através da variação do sigma da gaussiana. Com o objetivo de analisar o desempenho da rede variando o sigma da gaussiana, foi implementada a rede descrita anteriormente variando o sigma de 0.1 a 1.0 com passos de 0.1. Com o sigma de 0.1, a gaussiana é bem fechada e o pesos para os endereços com distância de Hamming igual ou maior que 1 é 0, que é o mesmo comportamento da RAM. A figura 8 ilustra o comportamento da rede.

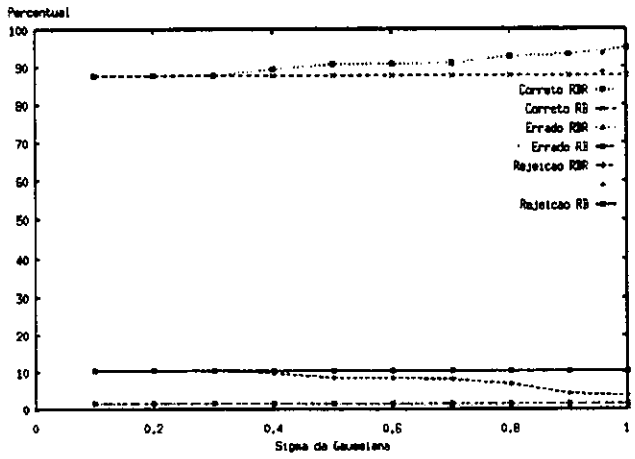


Figura 8: *O Desempenho de respostas corretas erradas e rejeitadascom variação dos pesos*

Os resultados obtidos mostram que com o aumento do sigma, aumenta a diferença de desempenho, com o RAM Radial tendo melhor desempenho de respostas corretas. Com esse aumento, houve uma diminuição de respostas erradas e rejeitadas.

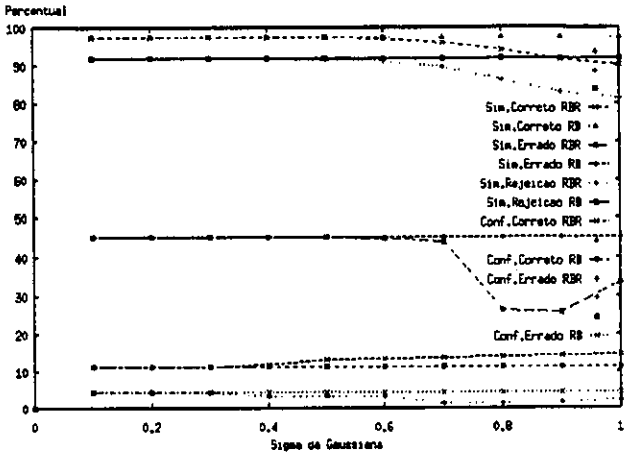


Figura 9: *O Desempenho da similaridade e confiabilidade com variação dos pesos*

A primeira vista, o melhor desempenho seria a rede com sigma igual a 1.0, mas ao observar o desempenho de similaridade e confiabilidade, figura 9, pode-se notar que a confiabilidade diminui com a diminuição do sigma mas a similaridade do RAM Radial vai diminuindo com o aumento do sigma, tendo uma diferença razoável com sigma igual a 1.0, em favor do RAM. Com isso, é mais adequado escolher um sigma menor pois tem uma melhor similaridade e também é confiável.

4.5 O Raio da Gaussiana

O raio da gaussiana define a esfera de distância de endereço da região de acesso. Com o objetivo de analisar o desempenho da rede variando o raio da gaussiana, foi implementada a rede descrita anteriormente variando o raio de 1, 2 e 3. Para avaliar estes experimentos, foi utilizado um sigma de 1.0 pois se o sigma for menor, o peso para distância de Hamming igual a 3 seria muito pequeno.

Ao verificar os resultados, figura 10, pode-se notar que o desempenho da rede é inversamente proporcional ao raio, ou seja, o melhor desempenho é obtida com um raio da gaussiana de 1. Isto ocorre pois com o raio igual a 1, a região de acesso tem um tamanho. Com o aumento do raio, há um aumento considerável no tamanho da região de acesso e ocorre a inclusão de endereços não muito similares a entrada, e isso dificulta a resposta dos neurônios.

Ao observar o desempenho da rede em relação a similaridade e confiabilidade, figura 11, pode-se ver que os melhores resultados são obtidos com raio igual a 1 pois a rede é mais confiável e possui as similaridades

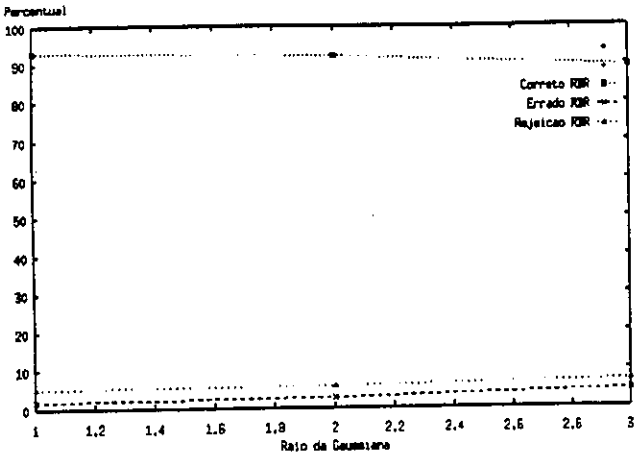


Figura 10: *O desempenho de respostas corretas erradas e rejeitadas com variação do raio da gaussiana*

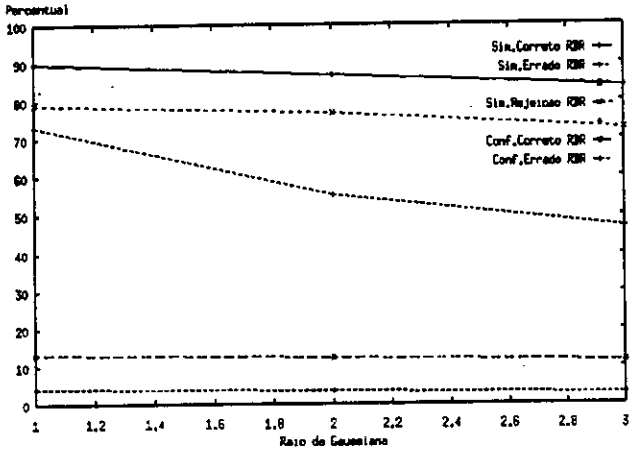


Figura 11: *O Desempenho da similaridade e confiabilidade com variação do raio da gaussiana*

mais próximas das da RAM. Novamente, esta queda ocorre devido a um aumento da região de acesso.

Naturalmente, é observado neste modelo que o uso de conectividades altas, tais como 10, 30 e 50, e conjuntos de treinamento alto, as redes com raios maiores teriam melhores resultados pois com o aumento exponencial na quantidade de endereços há uma necessidade de aumentar a região de acesso e o conjunto de treinamento. Com raios maiores, essa região aumenta e se terá melhores resultados.

5 Conclusão

O modelo proposto neste artigo é uma generalização do RAM original. Os modelos diferem na forma de acesso ao conteúdo. O modelo RAM Radial não acessa apenas um endereço para calcular a saída, mas sim toda uma região radial deste endereço

Para os experimentos realizados, a arquitetura utilizada é uma rede de discriminadores formada por apenas uma camada de neurônios.

A análise de desempenho mostrou que em todos os parâmetros, o RAM RAdial obteve melhores reconhecimento de resposta corretas que o RAM. O RAM Radial possui uma menor similaridade mas sempre **mais confiável** que o RAM. Um dos parâmetros que ocorreu uma melhora sensível no RAM Radial foi o conjunto de treinamento, com o conjunto de treinamento igual a 64, o desempenho da RAM Radial foi cerca de 93% de respostas corretas, enquanto o RAM original foi cerca de 89%. Estes resultados são interessantes para a construção de sistemas de reconhecimento de padrões, pois muitos sistemas que são baseados no modelo RAM e o método da n-tupla tem sido utilizado na prática e o modelo proposto comporta-se como uma versão otimizada de tais técnicas..

Referências

[1] E C B C Filho. *Investigation of Boolean Neural Network Based on a Novel Goal-seeking Neuron.* Doctor's thesis 1990.

[2] E C B C Filho, M C FairHurst and D L Bisset. *Adaptative Recognition using Goal Seeking Neurons*, Pattern Recognition Letters, 1991, 12, pp.131-138.

[3] I Aleksander. *Adaptative Systems of Logic Network and Binary Memories*, Proc. Spring Computer Conference. 1967.

[4] I Aleksander and H. Morton. *An Introduction to Neural Computing.* primeira edição. 1990.

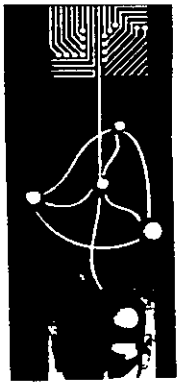
[5] I Aleksander, T J Stoneham. *A Guide to Pattern Recognition Using Random Acess Memories.* IEE Proc on Computer and Digital Techniques, vol.2, pp 29-36. 1979.

[6] I Aleksander, W V Thomas and P A Bowden. *Wisard - A Radical step forward in image recognition.* Sansor review, 4(3), pp 120-124, 1984.

[7] T Kohonen and H Ritter. *Self-organizing Semantic Maps*, Journal of Biological Cybernetics, Vol.61, pp. 241-254. 1989.

[8] R Linsker. *Self-organization in Perceptual Networks.* IEEE Journal of computer, Vol.61, number 3, pp.105-117, 1988

[9] W.W. Bledsoe and I. Browning. *Pattern recognition by machine.* The Eastern joint computer conference. pp. 225-232. 1959.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

Uso Cooperativo de Goal Seeking Neurons e Adaptive Logic Networks

W Martins and N M Allinson
Image Engineering Group, Electronics
University of York
York, United Kingdom

Resumo

A implementação física de Redes Neurais ainda é um dos grandes obstáculos para a sua utilização prática. Resultados significativos oriundos da cooperação entre dois modelos de fácil e econômica implementação reforçam a viabilidade de modelos que independem de variação de sinapses. A idéia básica deste artigo é o uso do treinamento iterativo de ALN em redes derivadas do treinamento (não-iterativo) de GSN.

1 Introdução

A popularidade de modelos cuja aprendizagem é desvinculada de variação de sinapses analógicas vem crescendo sensivelmente. O baixo custo e a facilidade da implementação física são fatores fundamentais neste fato. A deficiência de tais modelos está localizada freqüentemente no desempenho final. Alguns melhoramentos têm sido propostos para modelos específicos [1, 2, 3] com algumas exceções [4]. Este artigo apresenta resultados da combinação entre redes feed-forward de Goal Seeking Neurons [5] e Adaptive Logic Networks [6] e o algoritmo de conversão de topologias requerido.

A combinação entre os dois modelos envolvidos é feita através do uso de GSN como gerador de topologias a serem ré-treinadas pelo algoritmo iterativo de ALN. A aprendizagem em redes feed-forward de GSN é extremamente rápida, porque utiliza cada exemplo somente uma única vez. Em outras palavras, o treinamento é não iterativo. Além disso, existe um claro privilégio a exemplos utilizados no início do treinamento porque não se admite que uma nova informação destrua outra já registrada. Quando um exemplo gera uma modificação interna em um neurônio, esta modificação é irrevogável. Os autores já demonstraram que o controle dinâmico da ordem de apresentação de examplos é uma alternativa prática para resolver problemas onde várias redes são empregadas [7]. Neste artigo, resultados ainda melhores foram obtidos através do uso cooperativo de GSN e ALN.

Além do efeito positivo final na performance de classificadores empregados no reconhecimento de dígitos manuscritos, o produto final, redes de ALN, possui implementação (tanto em software quanto em hardware) mais fácil e econômica que a requerida por redes feed-forward de GSN. Por outro lado, o uso de redes de GSN resolve problemas de condições iniciais sofridos por ALN, pois possui maior funcionalidade a nível neuronal.

2 Breve Descrição Crítica dos Modelos

A nível neuronal, GSN são capazes de implementar qualquer função lógica, pois armazenam explicitamente a saída para cada possível situação de entrada. Neurônios de ALN, todavia, estão restritos às funções crescentes de duas variáveis: AND, OR,

LEFT e RIGHT (onde RIGHT(x,y) = y e LEFT(x,y) = x). Ao contrário dos neurônios de ALN, GSN possuem um terceiro estado 'u' para assinalar posições internas ainda indefinidas e podem ter mais que duas entradas. O aumento do número de entradas é, contudo, acompanhado da necessidade de mais posições internas, obedecendo lei exponencial (2^n, onde n é o número de entradas[1]). Apesar da melhoria no grau de saturação que o uso de mais posições internas resulta [8], este recurso é inconveniente sob ponto de vista econômico. Por outro lado, o estado final de neurônios de ALN ou são funções booleanas básicas, AND e OR, ou implementam simples cortes na topologia, RIGHT e LEFT. Ambos modelos utilizam uma única saída por neurônio, levando o uso de pirâmides como topologia natural (ALN usam árvores binárias mais especificamente). Novas abordagens foram recentemente publicadas para ultrapassar este obstáculo no caso de redes de GSN [2,3].

As limitações impostas na funcionalidade do neurônio em ALN são justificadas pelo aumento da insensibilidade das pirâmides a mudanças no nível de entrada [9], isto é, a saída final da rede não muda facilmente com pequenas variações na entrada. Como conseqüência, a implementação em hardware requer um simples circuito combinacional Booleano e a simulação por software pode ser acelerada sensivelmente através de *lazy evaluation*, já que tanto AND quanto OR podem ter a saída totalmente determinada pelo particular estado de uma única entrada.

Para aumentar a capacidade de mapeamento de ALN, entradas podem ser tomadas na sua forma normal ou complementada (previamente invertida). Mesmo assim, a dificuldade de implementar mapeamentos não-monotônicos pode ser demonstrada pelo valor 0,26% para a probabilidade de geração de topologias capazes de aprender a função XOR(x,y), assumindo eventos equiprováveis e quatro entradas (como aconselha o criador do modelo). A bem da verdade, ALN foram projetadas para implementar apenas funções monotônicas e esta característica é, às vezes, conveniente (como no controle de próteses para deficientes físicos [10], controle ativo de suspensão de automóveis [11]), pois possibilita a imposição de limites de segurança nas respostas das redes quando problemas com entradas e/ou saídas analógicas são enfrentados.

Assim como '0' e '1', o valor indefinido 'u' dos GSN pode ser armazenado, recebido e transmitido. Quando recebido, o valor indefinido acessa várias posições internas pois é entendido como '0' e '1' simultaneamente. Além de aproveitar melhor as posições internas já modificadas e, conseqüentemente, atrasar a ocorrência de saturação, a inclusão deste novo valor (com a mesma liberdade dos valores Booleanos originais) e a adoção de procedimentos determinísticos para manipulá-lo livra o sistema do caráter estocástico de seu predecessor. Os pontos negativos desta estratégia, além do maior número de bits requerido no estágio final, são o tempo gasto/complexidade final do chip para lidar com este valor indefinido no caso do uso de software/hardware e a necessidade de enviar dois bits pelos canais que interligam GSN no caso de implementação em hardware. O uso de redes de transputers é, portanto, adequado à implementação de redes de GSN.

Apesar de concordarem na escolha aleatória das topologias iniciais, as formas de treinamento adotadas por cada modelo são opostas: redes feed-forward de GSN são treinadas rapidamente de forma não-iterativa enquanto ALN procuram iterativamente um ponto de equilíbrio na aprendizagem de todos os exemplos. Enquanto redes de GSN mantêm a ordem de apresentação dos exemplos mesmo quando várias redes são utilizadas (como em classificadores), ALN modicam-na aleatoriamente antes de cada ciclo de

[1] Na verdade, a implementação do terceiro estado de GSN faz com que cada posição interna tenha pelo menos dois bits. O requerimento em bits para um neurônio de n entradas é, portanto, 2^{n+1}.

aprendizagem para evitar efeitos temporais (que prejudicam redes feed-forward de GSN).

GSN apresentam três estados: validação, aprendizagem e avaliação. Um exemplo é aprendido se, e somente se, passa com sucesso pela fase de validação. Durante validação, neurônios são estimulados a responderem 'u'. Isto acontece quando não apenas um único valor Booleano é acessado. Se o neurônio da última camada responder 'u' ou o valor que se deseja ensinar, o exemplo foi validado com sucesso, isto é, um mapeamento entre o nível de entrada e o nível de saída pode ser implementado para cumprir o objetivo desejado. Na fase de aprendizagem, o neurônio irá modificar posições internas (se necessário). Durante o treinamento, a validação é feita através da propagação feed-forward de estímulos. Em caso positivo, os neurônios de cada nível entram em estado de aprendizagem logo após o mesmo processo ter terminado para o nível seguinte, já que precisam saber qual é o específico mapeamento requerido. A dinâmica é similar, portanto, ao algoritmo de back-propagation: uma fase feed-forward de propagação de estímulos seguida de outra fase em sentido contrário para ajustes na topologia. Durante avaliação, o procedimento típico é o uso do valor Booleano mais frequente dentre as posições acessadas.

O algoritmo de aprendizagem de ALN é, a nível neuronal, baseado na freqüência de acesso, no sentido de respeitar a maioria das solicitações de mapeamento e destina-se a determinar a saída quando as entradas têm valores Booleanos diferentes. Um contador interno para cada entrada e um threshold são empregados durante treinamento. Se o contador está acima do threshold e a entrada associada é '1', a saída será também '1'. Para os casos onde as entradas são idênticas, a saída tem o mesmo valor e é independente de contadores. No início do treinamento, os contadores são aleatoriamente pré-definidos com valores próximos ao threshold para que possam mudar de comportamento rapidamente. A nível global da topologia, nem todos os neurônios têm contadores atualizados a cada apresentação de um exemplo. A atualização, a exemplo das redes de GSN, também começa no nível de saída para mover em direção ao nível de entrada, mas é interrompida quando se encontra um neurônio cujas entradas têm valor idêntico ao da saída final. Ao contrário do algoritmo original de GSN, ALN impõe direta correlação entre saída de neurônios e saída final. Este é o ponto explorado pelos anteriormente citados novos algoritmos de treinamento para GSN que possibilitam o uso de topologias não-piramidais.

3 Vantagens do Uso Cooperativo

As vantagens do uso do algoritmo de ALN em árvores binárias oriundas da conversão de redes de GSN são claras. A única desvantagem, aumento do período de treinamento, é fortemente justificada pela melhor performance e redução na complexidade da implementação finais.

Devido à usual forma de escolher topologias (árvores binárias com entradas randomicamente selecionadas em forma normal/complementada), a probabilidade de ALN descobrirem relações não-monotônicas é virtualmente zero (particularmente em níveis superiores). Redes de GSN podem mostrar quais relações são necessárias e passar este conhecimento para ALN. GSN adotam entradas em forma complementada sem maiores esforços conforme a necessidade dos mapeamentos. ALN sofrem quando o número de entradas é pequeno porque necessitam descartar todas as subárvores que não foram escolhidas numa forma admissível. A solução é aumentar o tamanho da rede, contrariando a busca de baixo custo e alta eficiência. Por fim, o fato de que GSN podem utilizar neurônios com mais de duas entradas pode ser utilizado para descobrir relações dificilmente encontradas por ALN. Sob ponto de vista das redes feed-forward de GSN, o algoritmo iterativo de ALN corrige as distorções geradas pela ordem de apresentação dos exemplos.

4 Conversão de GSN para ALN

Primeiramente, suponha que, no final do treinamento, todas as posições internas dos neurônios que compõem a rede de GSN contêm um dos valores Booleanos, '0' ou '1'. Neste estado, a simples aplicação de Álgebra Booleana (em particular, dos Teoremas de Morgan), a partir do neurônio do nível de saída em direção ao nível de entrada, é suficiente para resultar numa topologia equivalente onde todas as inversões (uso de NOT) estão aplicadas às entradas e todos os neurônios comportam-se como AND ou OR (Fig. 1). A modificação para que todos neurônios tenham apenas duas entradas é trivial através da inclusão de novos neurônios. A topologia equivalente gerada já obedece o formato de ALN.

Normalmente, contudo, várias posições permanecem indefinidas durante todo o treinamento. Como ALN não trabalham com este tipo de valor, todas posições indefinidas devem ser modificadas de modo a que o caso se reduza ao anteriormente descrito. No caso de GSN com duas entradas, a preocupação por aumentar o poder discriminatório do neurônio justifica a busca do equilíbrio entre a freqüência de utilização de ambos valores Booleanos. No caso de GSN com mais entradas, estratégias que analisam os vizinhos de cada posição indefinida são plausíveis. Este último caso será reportado em um futuro artigo. Será interessante notar o efeito final causado pelo descobrimento de relações que envolvem mais do que duas variáveis.

5 Trabalho Experimental e Resultados

Muitas simulações foram conduzidas para testar a validade das idéias anteriores. Cada resultado da tabela 1 é a média de 100 simulações. De modo a fazer boas comparações, topologias iniciais idênticas (randomicamente escolhidas) foram utilizadas para as versões originais e combinada. Redes menores foram utilizadas como componentes de modo que redes de 64 entradas são resultado do acoplamento de novas redes de 32 entradas em redes pré-existentes. Esta estratégia parece apropriada, pois é usual a necessidade de saber quanto melhor uma rede se torna quando mais entradas são utilizadas.

O conjunto de exemplos é composto de 160 imagens preto-branco (16 por 16 pixels) de dígitos manuscritos. Para avaliar, 800 padrões foram empregados. A tarefa não é simples porque várias superposições entre classes acontece no espaço de entrada. Nenhum pré-processamento foi realizado, as variáveis de entradas são os pixels. Os dados foram coletados através de um digitalizador onde pequenos quadrados indicavam onde o usuário deveria escrever. Para códigos de saída, empregou-se a codificação de Hadamard, ou seja, códigos de comprimento igual a 15 bits onde cada par de códigos mantém uma distância Hamming de 8 bits entre seus componentes. O número de entrada por rede é de 32, 64 ou 128. O número de redes por cada bit dos códigos de saída é 3, 5 ou 11. Todas entradas foram randomicamente associadas às posições dos níveis de entrada das redes. Nenhum uso da natureza bidimensional da tarefa foi empregado. Os autores acreditam, portanto, que a mesmo método aqui descrito seja adequado para outras tarefas da área de reconhecimento de padrões.

Desempenhos de k-NN classificadores podem ser vistos na tabela 2. Nenhum outro tipo de classificador possui treinamento mais rápido, pois consiste simplesmente em armazenar os exemplos disponíveis. A avaliação, contudo, é lenta porque as distâncias entre o padrão corrente e todos os exemplos armazenados devem ser computadas de modo a revelar qual é a opinião da maioria dos k vizinhos mais próximos. Por outro lado, com relação a taxa de reconhecimento, estes classificadores são bem reputados.

A superioridade do sistema resultante da combinação entre os dois modelos é mostrada pelos resultados (veja tabela 1). É interessante notar como o desempenho das redes de GSN afeta o sistema final. Além disso, a taxa de reconhecimento final do

sistema em sua versão mais forte, que utiliza mais redes e onde as redes têm maior capacidade, é comparável ao melhor desempenho de classificadores *k*-NN. Convém lembrar que, devido ao fato de que o grau de robustez nestes classificadores é diretamente proporcional ao número de vizinhos consultados, não é usual a utilização de um único vizinho.

6 Conclusão

Este artigo apresentou e comparou dois modelos de redes neurais onde a aprendizagem é desvinculada da variação de sinapses: redes feed-forward de Goal Seeking Neurons e Adaptive Logic Networks. Demonstrou-se, através de trabalho experimental (reconhecimento de dígitos manuscritos), como a aplicação do treinamento iterativo pertencente a ALN pode melhorar o desempenho de topologias derivadas de redes de GSN.

As idéias principais do procedimento de conversão de redes de GSN para o formato de ALN foram suscintamente descritas.

O sistema resultante da cooperação entre os dois modelos mostrou-se nitidamente superior. Além do desempenho, vários outros pontos respaldam a estratégia desenvolvida sob ponto de vista de ambos modelos envolvidos, destacando-se: a descoberta de relações não-monotônicas e inversões de variáveis de entrada para ALN e a redução da complexidade final (implicando em menor custo e maior velocidade) para GSN.

Acknowledgement

W Martins goza de licença da Universidade Federal de Goiás e é bolsista do CNPq em fase de finalização de doutoramento na Universidade de York, Inglaterra.

Referências

[1] Bowmaker, R G, and Coghill, G G, 1992, "Improved Recognition Capabilities for Goal Seeking Neuron", in *Electronic Letters*, vol. 28, No. 3, 220-221.

[2] Martins, W and Allinson, N M, 1993, "Two improvements for GSN neural networks", in *Proc. Weightless Neural Network Workshop '93*, 58-63, York, UK.

[3] Carvalho, A de et al, 1994, "Progressive learning algorithm for GSN feedforward neural architectures", *Electronic Letters*, vol. 30, No. 6, 506-507.

[4] Muselli, M, 1994, "Hamming clustering: improving generalization in binary neural networks", *Proc. of the Intl Conf on Artifcial Neural Networks*, vol. 2, 1083-1086, Sorrento, Itália.

[5] Bisset, D L et al, 1989, "A comparative study of neural network structures for practical application in a pattern recognition environment" in *Proc. First IEE Interl. Conf. on Artificial Neural Networks*, 378-382, London, UK.

[6] Armstrong, W W and Gecsei, J, 1979, "Adaptation Algorithms for Binary Tree Networks", *IEEE Trans on System, Man and Cybernetics*, vol. 9, 276-285.

[7] Martins, W and Allinson, N M, 1994, "Improving GSN-based classifiers by control of presentation order", in *Proc. do I Simpósio Brasileiro de Redes Neurais*, Caxambu, MG, artigo aceito.

[8] Filho, E, 1990, "Investigation of boolean neural network based on a novel Goal-Seeking-Neuron", PhD Thesis, Electronic Engineering Laboratories, University of Kent, UK.

[9] Armstrong, W W and Bochmann, G v, 1974, "Properties of Boolean Functions with a Tree Decomposition", *BIT*, vol. 13, 1-13.

[10] Stein, R B et al, 1992, "Methods to Control Functional Electrical Simulation in Walking", *Proc. of the First Intl FES Symposium*, Sendai, Japan, 135-140.

[11] Armstrong, W W and Thomas, M M, 1994, "Control of a Vehicle Active Suspension System Model using Adaptive Logic Networks", *Program Addendum of the World Congress on Neural Networks*, San Diego, USA, 9-14.

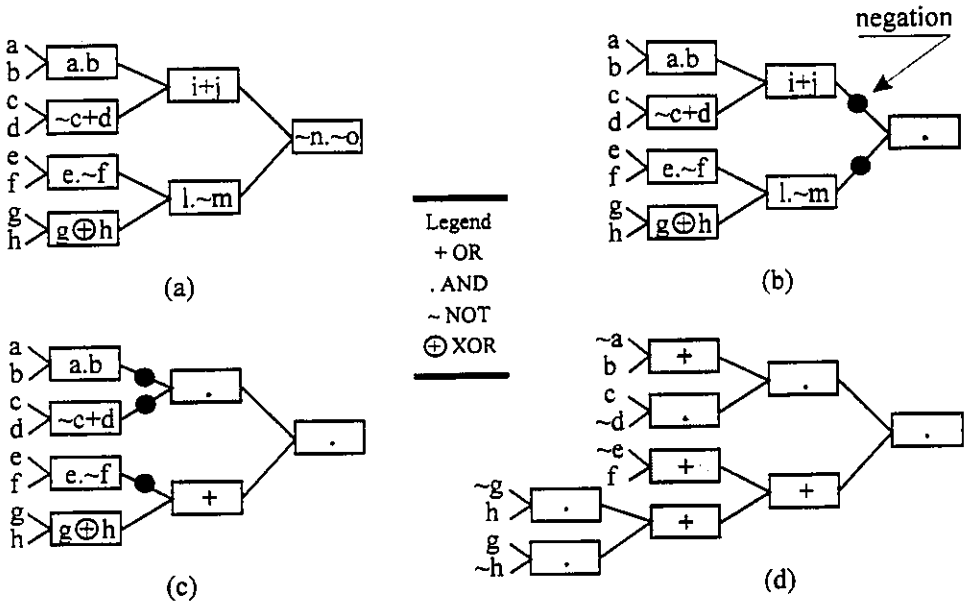


Figura 1: Exemplo de conversão de GSN para ALN: (a) estado inicial; (b) conversão do nível de saída; (c) conversão de nível intermediário; (d) estado final.

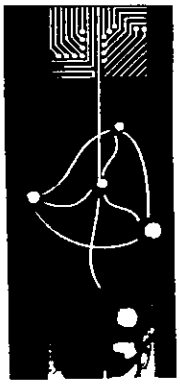
TREINAMENTO			*SITUAÇÃO*	*AVALIAÇÃO*		
redes per bit/entradas per rede				*redes per bit/entradas per rede*		
3 / 32	**5 / 32**	**11 / 32**		**3 / 32**	**5 / 32**	**11 / 32**
53.59	63.43	74.80	*original GSN*	39.57	47.21	56.95
73.46	80.76	85.08	*original ALN*	57.98	66.76	74.20
80.42	89.36	96.70	*combinação*	61.19	71.40	83.78
3 / 64	**5 / 64**	**11 / 64**		**3 / 64**	**5 / 64**	**11 / 64**
59.04	68.84	80.63	*original GSN*	43.78	52.25	64.35
81.40	84.79	87.76	*original ALN*	67.57	72.85	77.32
91.84	96.02	98.78	*combinação*	74.83	82.48	88.76
3 / 128	**5 / 128**	**11 / 128**		**3 / 128**	**5 / 128**	**11 / 128**
66.34	75.49	85.49	*original GSN*	47.57	56.69	68.65
85.12	87.68	88.94	*original ALN*	70.26	74.48	77.09
97.21	98.71	99.51	*combinação*	82.03	87.25	90.53

Tabela 2: Taxa de reconhecimento para o conjunto de exemplos do treinamento e o conjunto de padrões de teste.

# vizinhos k	reconhecimento treinamento (%)	reconhecimento avaliação (%)
1	100.00	91.50
2	96.88	84.75
3	91.88	79.88
4	92.50	81.75
5	93.13	78.00
15	85.63	69.87

Tabela 2: Performances de classificadores k-NN.

Anotações

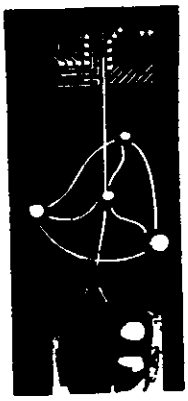


1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

ALGORITMOS DE TREINAMENTO

Anotações



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajuba
Itajuba, 24 a 27 de outubro de 1994

UM ALGORÍTIMO DE TREINAMENTO CONTÍNUO PARA REDES MULTI CAMADAS

C.E. Pedreira e N.M.Roehl
Dept. de Eng. Elétrica PUC-Rio
C.P. 38063
22452-970 Rio de Janeiro
pedreira@ele.puc-rio.br

Resumo

Neste artigo se propõe um novo procedimento para treinamento contínuo de redes em camadas. Após a etapa tradicional de treinamento o algorítimo adaptativo ajusta, se e somente se necessário o conjunto de pesos, permitindo assim uma adaptação a um possível novo modelo. Trata-se portanto de metodologia especialmente interessante para sistemas variantes no tempo. Ao projetista é reservada em cada momento a decisão de que tolerância sobre o erro do último dado se está disposto a permitir. Tem-se um problema de compromisso entre casar o dado novo e perturbar o mínimo possível o conjuto de pesos gerado pelo treinamento anterior.

INTRODUÇÃO

Diversas aplicações de grande importância involvem sistemas variantes no tempo. A dificuldade intrínsica em lidar com esses tipos de sistemas induz muitas vezes à introdução de hipóteses artificiais que podem ser causadoras de resultados não satisfatórios. Redes neurais em camadas demonstraram-se uma ferramenta importante em uma variedade de problemas quando a hipótese de invariância do sistema é aceitável. Nossa contribuição consiste em propor um novo algoritmo de ajuste de pesos em redes multi camadas que se mostra conveniente à modelagem de sistemas variantes no tempo.
O principal objetivo é manter o erro relacionado ao dado novo dentro de uma tolerância pré estabelecida, minimizando a perda da informação incorporada pelo treinamento com os dados originais. Dessa forma, ao projetista é permitado julgar a relevância que se deseja atribuir à nova informação. Esforço neste sentido foi anteriormente feito por Park et al. [1], sem esta flexibilidade. Em [1], o último dado é obrigado a ter erro zero, em detrimento do treinamento anteriormente realizado. Esta falta de flexibilidade é especialmente pouco desejável em situações onde sabe-se de ante mão da possibilidade de mudanças bruscas, ou quando o sistema retorna ao modelo anterior com razoável rapidez. No primeiro caso, parece mais razoável informar a rede desta incerteza pré conhecida e controlar assim o erro que se pode permitir. No segundo, a metodologia proposta em [1] irá causar fortes danos ao conjuto de pesos original dificultando, deste modo, um retorno ao modelo original.

O PROBLEMA

Seja uma rede com uma camada oculta descrita da seguinte forma:

$$y = f_1[V^t u] \quad u = f_2[W^t x] \quad (1)$$

onde $y \in \Re^o$ é a saída da rede, $u \in \Re^h$ é o vetor de ativações da camada escondida e $x \in \Re^I$ é o vetor de dados de entrada. As

matrizes W e V contêm os pesos das ligações entre camadas de entrada e escondida e camadas escondida e de saída, respectivamente. As funções vetoriais $\mathbf{f_1} \in \Re^o$ e $\mathbf{f_2} \in \Re^h$ são tais que $f_1^i(.)$ e $f_2^i(.)$ são funções diferenciáveis não decrescentes. Para simplificar o desenvolvimento, vamos supor as funções sigmoidais. Essa formulação pode ser facilmente generalizada para redes com mais de uma camada embituda.

A partir de uma rede treinada, i.e., um conjunto de pesos (W,V) apropriado para os pares entrada-saída originais $((x_i,d_i)$, i = 1,...,n) e um dado novo (x_{n+1},d_{n+1}), o objetivo é determinar um novo conjunto de pesos (W',V') tal que a seguinte função energia é minimizada, sujeita à nova restrição imposta pelo último dado apresentado.

$$\text{Min } E = 1/2 \sum_{i=1}^{N+1} |d_i - y_i|^2 \qquad (2)$$

t.q.

$$-\varepsilon^l \le (d^l_{n+1} - y^l_{n+1}) \le \varepsilon^l,\ l = 1,...,o \qquad (3)$$

onde $\varepsilon^l \in (0, \min(d^l_{n+1}, 1- d^l_{n+1}), l = 1,...,o$ é a tolerância associada à nova restrição e $|.|$ é a norma Euclidiana. Pode-se provar que estes limites impostos sobre ε garantem a existência da função inversa $(f_1)^{-1}$.

A função objetivo (2) reflete o desejo de minimizar o erro relativo aos dados de treinamento originais enquanto a restrição (3) manterá o erro associado ao último dado dentro de uma tolerância pré estabelecida ε. Isso significa, que se está lidando com uma solução de compromisso entre a informação do sistema antigo e o dado novo, possivelmente refletindo uma variação na planta original. Essa abordagem permite ao projetista controlar a tolerância, ou seja, decidir o quão relevante o dado novo é comparado aos dados anteriormente utilizados na identificação do sistema.

De modo a simplificar o tratamento analítico e computacional do problema P_β é preciso linearizar a restrição (3). Desse modo, suponha-se que o novo conjunto (W',V') pode ser escrito como: V' = V+ΔV e W' = W+ΔW, onde os incrementos $\Delta V \in \Re^{hxo}$ e $\Delta W \in \Re^{Ixh}$. De modo a simplificar a notação, os índices de super escritos serão omitidos. Então, para (x_{n+1},d_{n+1}) das equações (1) e (3) temos

$$-\varepsilon \le d_{n+1} - f_1[(V+\Delta V)^t u] \le \varepsilon$$

logo,

$$d_{n+1} - \varepsilon \le f_1[(V+\Delta V)^t u] \le d_{n+1} + \varepsilon.$$

Como $(f_1^i)^{-1}$ é uma função monotonamente crescente $\forall$ i = 1,...,o:

$$(d_{n+1} - \varepsilon) \le (V+\Delta V)^t u \le f_1^{-1}(d_{n+1} + \varepsilon) \quad (4)$$

Por outro lado, aplicando-se a expansão de Taylor de primeira ordem obtem-se:

$$u \approx f_2(W^t x_{n+1}) + J f_2(W^t x_{n+1})\, \Delta W^t x_{n+1}. \qquad (5)$$

E, substituindo-se a aproximação (5) na desigualdade (4), chega-se a:

$$f_1^{-1}(d_{n+1} - \varepsilon) + V^t f_2(W^t x_{n+1}) \le$$

$$\le V^t J f_2(W^t x_{n+1})\, \Delta W^t x_{n+1} +$$

$$+ \Delta V^t f_2(W^t x_{n+1})$$

$$\le f_1^{-1}(d_{n+1} + \varepsilon) - V^t f_2(W^t x_{n+1}) \qquad (6)$$

Observe-se que essa aproximação é válida se e somente se H << 1, onde H é o Hessiano de $f_2(W^t x_{n+1})$, e $\Delta V_{ij} << V_{ij}$, $\forall$ i = 1...h e j = 1...o. Ao limitar as pertubações sobre W e V, definine-se uma região onde a hipótese de linearidade é válida. Rearrumando-se ΔV^t e Δ W em forma vetorial, através do operador vec, a restrição (6) pode ser escrita numa forma compacta como:

$$c_1^i(\varepsilon) \le (Az)^i \le c_2^i(\varepsilon) \quad i = 1...o \qquad (7)$$

onde $A = [A_p : A_q]$, $A \in \Re^{o(p+q)}$

$p = Ixh$ e $q = hxo$

$z = [\Delta W_{vec}{}^t : (\Delta V^t)_{vec}{}^t]^t$, $z \in \Re^{(p+q)}$

$A_p \in \Re^{oxp}$ é a solução do sistema de equações:

$$A_p \Delta W_{vec} = V^t J f_2(W^t x_{n+1}) \Delta W^t x_{n+1}$$

$A_q = [A_i] \in \Re^{oxq}$ onde

$A_i = [0...0\ f_2(W^t x_{n+1})^t\ 0...0]$, $i = 1...o$

$$c_1(\varepsilon) \equiv f_1^{-1}(d_{n+1} - \varepsilon) + V^t f_2(W^t x_{n+1}),$$

$c_1(\varepsilon) \in \Re^o$

$$c_2(\varepsilon) \equiv f_1^{-1}(d_{n+1} + \varepsilon) - V^t f_2(W^t x_{n+1}),$$

$c_2(\varepsilon) \in \Re^o$.

A região factível pode então ser definida como a intersecção entre a região de linearidade e o hiperespaço definido pela restrição (7).

A função energia pode também ser reescrita numa forma compacta como

$$J(z) = 1/2\ z^t K z$$

onde $K = S^tS$ e S é a matriz de sensibilidade de y às pequenas variações nos pesos, obtida após algum algebrismo. A variação de energia associada aos N+1 dados devido a variações nos pesos pode ser escrita como

$$\Delta E = [\Delta E_1 \ldots \Delta E_{N+1}]^t = Sz.$$

Após rearrumar (2) em termos de variação de energia, chega-se a:

$$J = 1/2 \sum_{i=1}^{N+1} (E_{i,W} - E_{i,W'})^2$$

onde $E_{i,W}$ e $E_{i,W'}$ são os erros de saída relacionados ao i-ésimo dado com osconjuntos de pesos (W,V) e (W',V'), respectivamente. Logo, tem-se que

$$J = 1/2 \sum_{i=1}^{N+1} \Delta E_i^2 = 1/2\ \Delta E^t\ \Delta E = 1/2\ z^t K z$$

O problema P_ε pode agora ser formulado da seguinte forma:

(P_β) Min $J(z) = 1/2\ z^t K z$

s.a $c_1(\varepsilon) \le Az \le c_2(\varepsilon)$

$z \in \beta$

onde $\beta = \{z \in \Re^{p+q}: |z_i| \le \delta_i,\ i = 1,...,p+q\}$, $\delta_i > 0$ é a região de linearidade da restrição (3). Utilizando-se propriedades da função inversa de f_1, a desigualdade de Shwartz e um valor arbitrariamente pequeno para o erro da aproximação linear de Taylor, determina-se valores bem definidos para δ_i, para todo i = 1,...,p+q.

Observa-se que simplificando-se a rede para uma única unidade de saída e ajustando-se $\varepsilon = 0$, a restrição (7) se tornará idêntica à restrição imposta em [1]. Dessa forma, P_β é de fato, uma generalização do problema proposto em [1], onde uma restrição de igualdade $d_{n+1} = f_1[(V+\Delta V)^t u]$ é utilizada para obrigar que a rede se ajuste perfeitamente ao último dado apresentado. Essa restrição é um hiperplano de simetria para o hiperespaço definido por (7), cuja largura é função do parâmetro ε (figura 1).

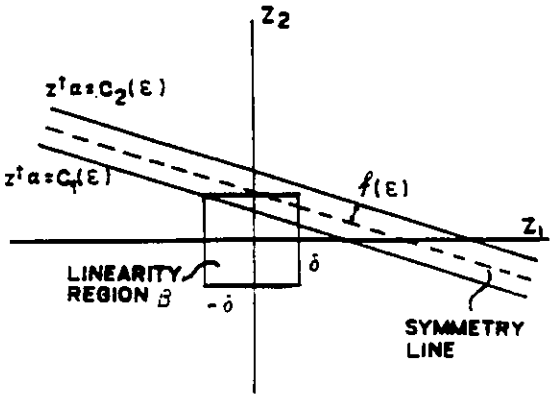


Figura 1

Outra característica introduzida nesse algoritmo é o tratamento de outliers nos dados. Foi implementada uma heurística simples que se baseia no erro de saída da rede e evita que os pesos se alterem caso haja uma alta probabilidade do dado novo se caracterizar com um outlier. Se, no entanto, os dados seguintes apresentarem o mesmo tipo de comportamento, interpreta-se que provavelmente estará ocorrendo uma mudança estrutural no sistema, devendo os pesos serem então modificados para captar essa nova tendência. Esse tipo de tratamento parece encontrar aplicação especialmente em problemas de previsão onde a ocorrência de outliers é frequente.

ANÁLISE DO PROBLEMA

Trata-se a seguir da análise teórica da questão de existência de solução para o problema P_β. Formulam-se duas proposições que estão relacionadas à essa análise, para o caso de uma única unidade de saída. Resultados análogos podem ser obtidos para o caso de redes com saídas múltiplas após algum algebrismo.

A proposição (P1) refere-se ao caso em que o hiperplano de simetria da região definida pela restrição (7) intercepta a região de linearidade β (figura 1). A proposição (P2) fornece às condições sobre o parâmetro ε que garantem a existência de solução quando esse hiperplano não intercepta β.

Proposição (P1)

Seja $r \equiv \{z: < z, a > = c\}$ onde $c = f_1^{-1}(d_{N+1}) - v^t f_2(W^t x_{N+1})$, o hiperplano de simetria da região definida pela restrição (7). Suponha-se que r intercepta a região de linearidade β, então o problema P_β tem solução para todo $\varepsilon \in (0, \min(d_{N+1}, (1-d_{N+1}))$.

Prova: cf. [2] e [3]

Proposição (P2)

Seja $\varepsilon_{max} = \min(d_{N+1}, (1-d_{N+1}))$. Assume-se que o hiperplano de simetria não intercepta a região de linearidade. Então, se $\varepsilon^* \leq \varepsilon_{max}$, o problema P_β tem solução $\forall \varepsilon \in [\varepsilon^*, \varepsilon_{max})$.

Prova: cf. [2] e [3]

Quando o hiperplano de simetria não intercepta a região de linearidade, o projetista deverá aumentar o valor do parâmetro ε de modo a garantir a existência de uma solução apropriada. Vale notar que se $\varepsilon > \varepsilon_{max}$ o problema não terá solução. Neste caso, ao usar o algorítimo proposto em [1] se estará violando fortemente a hipótese de linearidade, provocando assim um erro não previsível e não controlado.

A SOLUÇÃO DO PROBLEMA

O Problema P_β pode ser resolvido por uma variedade de algorítimos padrão de programação não linear. Embora, no caso geral, não se possa garantir otimização global para tais esquemas, no presente caso, tem-se convergência global uma vez que a função de custo J é uma função convexa sobre uma região viável também convexa [4]. Descreve-se a seguir o algorítimo proposto para solução do problema P_β. Um algorítimo do tipo projeção de gradiente [4] é usado para resolver o problema de programação quadrática que aparece no passo 4.

Algorítimo para solução de P_β :

Passo 0 - Verifique através de uma heurística se o dado novo tem alta probabilidade de ser um "outlier".

Passo 1 - Calcule a matriz de sensibilidade S (para maiores detalhes ver [2]), calcule $K \equiv S^t S$.

Passo 2 - Ache as restrições lineares

$$Az = c1 \quad e \quad Az = c2$$

Passo 3 - Ache o limitante β usando:

$$-\delta_i \leq z_i \leq \delta_i, \; i = 1...(p+q)$$

Passo 4 - Através do método de gradiente projetado ache z que minimiza a função de custo $J(z) = 1/2\ z^t\ K\ z$ sujeito às restrições $c_1(\varepsilon) \leq Az \leq c_2(\varepsilon)$ e $z \in \beta$.

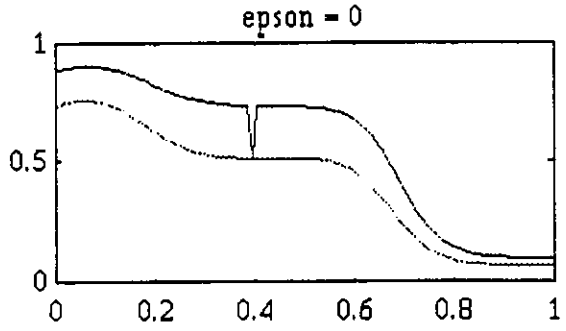
Passo 5 - Ache ΔW e ΔV a partir de z.

Passo 6 - Atualize W e V:

$$W' = W + \Delta W \text{ e } V' = V + \Delta V.$$

Uma característica importante do algorítimo proposto está relacionada com a não alteração dos pesos quando o dado novo não representa uma mudança do modelo. Deste modo, pode-se deixar o sistema adaptativo permanentemente ligado, com a garantia de que só serão feitas modificações em caso de necessidade. A prova desta propriedade se encontra em [2].

Como ilustração gráfica do algoritmo de treinamento contínuo proposto, apresentamos na figura 2 resultados relacionados a um exemplo simples de uma rede com uma unidade de entrada, uma unidade de saída e uma camada embutida. A rede foi inicialmente treinada através do algoritmo de retropropagação do erro para captar o mapeamento existente entre 100 pontos da curva. Provoca-se um desvio de 40% num dos pontos e aplica-se o algoritmo de treinamento contínuo para valores diferentes do parâmetro ε. Observa-se que a medida que o valor de ε aumenta, menor é o desvio da resposta da rede para os outros pontos da curva. Entretanto, pior é o ajuste para o dado novo.



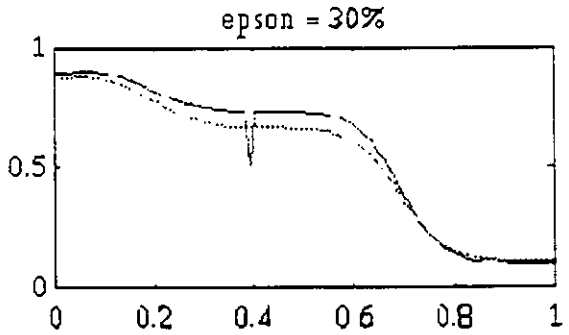


Figura 2

COMENTÁRIOS FINAIS

O esquema de treinamento contínuo aqui apresentado, permite que o projetista decida o quanto está disposto a pagar pela adaptação do dado novo. Esta metodologia parece ser bastante interessante especialmente quando se tem informação a respeito da velocidade de alteração do modelo, e também do nível de tolerância aceita pela aplicação em questão. Embora se tenha o campo das aplicações como meta, este artigo representa uma contribuição de cunho teórico. Pesquisa está no momento em andamento visando testar o desempenho numérico do algorítimo proposto, especialmente para previsão e classificação.

REFERÊNCIAS:

[1] Park D.C., El-Sharkawi M.A., e Marks II R.J., "An Adaptively Trained Neural Network", IEEE Trans. on Neural Networks, Vol 2, N. 3, Maio 1991.

[2] Pedreira C.E. e Roehl N.M., "On Adaptively Trained Neural Networks", Comunicação Interna CSC 36-93 PUC-Rio Dept. de Eng. Elétrica, Março de 1993.

[3] Pedreira C.E. e Roehl N.M., "On Adaptively Trained Neural Networks", Proceedings do 1993 Intern. Joint Conference on Neural Networks, Vol 1, pp 565-568, Nagoya, Japão, Outubro 1993.

[4] Gill P.E., Murray W. e Wright M.H., Practical Optimization, Academic Press 1981.

1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajuba, 24 a 27 de outubro de 1994

"STOCHASTIC PARAMETER ESTIMATION NEURAL NETS SUPERVISED LEARNING APPROACH"

Atair Rios Neto
Instituto Nacional de Pesquisas Espaciais-INPE
São José dos Campos, SP

ABSTRACT

In this short paper the problem of supervised learning of a neural net is treated as one of stochastic parameter estimation. The identification of the neural net parameters in its supervised training is as usually formulated as an optimization problem. A linear pertubation scheme, reducing the problem to one of stochastic parameter estimation in each iteration is then proposed. The structure of a Kalman filtering in the resulting algorithm allows the use of all the existing results in state and parameter estimation to guarantee a good performance to the proposed procedure.

1. INTRODUCTION

The experience with adaptative state estimation and control schemes based on the Kalman filter (e.g. Jazwinki, 1970) in aerospace applications (Rios Neto and Kuga, 1985; Rios Neto and Cruz, 1990) and in parameter estimation (Orlando and Rios Neto, 1984), has naturally led the author to consider these schemes in his recent work with neural nets in control.

Looking at the problem of neural nets supervised training in the modelling of nonlinear systems as one of parameter identification, and following an approach very much influenced by the one proposed by Chen and Billings (1992), this short paper tries to give an interdisciplinary contribution. It shows how to formulate and treat this problem as one of stochastic parameter estimation, with the objective of calling the attention of researchers in the field of neurocomputing for results of the area of state and parameter estimation that can be of great utility to them.

2. NEURAL NET SUPERVISED LEARNING APPROACH

The supervised training of a neural net to learn a nonlinear continuous mapping

$$f(.): x\epsilon D \subset R^{nI} \rightarrow y\epsilon R^{no} \quad ...(2.1)$$

can naturally be treated as a problem of estimation the connection weight parameters p in the network correspondent mapping:

$$\hat{f}(.,p): x\epsilon D \subset R^{nI} \rightarrow y\epsilon R^{no} \quad ..(2.2)$$

such as to have $\hat{f}(x, \bar{p})$ as close as possible to f(x) for $x\epsilon D$.

A set of pairs (x(t), y(t)), t=1,2,...N, given by the mapping in (2.1) is selected in order to get this approximation, and the parameters are usually determined under the condition of minimizing

$$J(p)=\tfrac{1}{2}((p-\bar{p})^T\bar{P}^{-1}(p-\bar{p})+ +\sum_{t=1}^{n} (y(t)-\hat{y}(t))^T R^{-1}(t)(y(t)-\hat{y}(t))) \quad ...(2.3)$$

where $\bar{p}$ is a given a priori value of p; $\hat{y}(t)= f(x(t),p)$; $\bar{P}^{-1}$ and $R^{-1}(t)$ are weight matrices.

To solve the problem of (2.3) an iterative scheme based on linear perturbation has to be used. In a typical iteration, one usually takes:

$$J(p^k)= \tfrac{1}{2}((p^k-\bar{p})^T\bar{P}^{-1}(p^k-\bar{p}) +\sum_{t=1}^{n}(\alpha^k(y(t)-\bar{y}^k(t))- \hat{f}_p(x(t),\bar{p}^k)(p^k-\bar{p}^k))^T R^{-1}(t) (\alpha^k(y(t)-\bar{y}^k(t))-\hat{f}_p(x(t),\bar{p}^k) (p^k-\bar{p}^k))) \quad ...(2.4)$$

where k=1,2,...,kc; $\bar{p}^1=\bar{p}$, $\bar{y}^k(t)=f(x(t),\bar{p}^k)$; $\hat{f}_p(x(t),\bar{p}^k)$ is the matrix of first partial derivatives with respect to p; $0<\alpha^k\leq 1$ is an adjusting parameter to guarantee the hypotesis of linear perturbation.

The solution of (2.4) is formally equivalent (see, for example, Jazwinki, 1970) to the following stochastic parameter estimation problem:

$$\bar{p}= p^k+\bar{e} \quad ...(2.5)$$

$$\alpha^k(y(t)-\bar{y}^k(t))= \hat{f}_p(x(t),\bar{p}^k) (p^k-\bar{p}^k)+v(t) \quad ...(2.6)$$

where, $E[\bar{e}]=0$, $E[\bar{e}\bar{e}^T]= \bar{P}$, $E[v(t)]=0$, $E[v(t)v^T(t)]= R(t)$, usually diagonal; E[.] is the expextation value operator; $\bar{e}$ and v(t) are assumed to be gaussian distributed and not correlated; and v(t) is also assumed not correlated along t=1,2,..., N.

Expressing this problem in a more compact notation, one has:

$$\bar{p}= p^k+\bar{e} \quad ...(2.7)$$

$$z^k= H^k p^k + v \quad ...(2.8)$$

where in (2.6) $\hat{f}_p(x(t), \bar{p}^k)\bar{p}^k$ has been taken to the left hand side and all the values of the index t were considered to define the extended z^k, H^k and v.

Thus, in a typical iteration, for k=1,2,..., kc, the resulting parameter estimation problem can then be solved with a Gauss Markov estimator, which in a Kalman form results as:

$$\tilde{p}^k = \tilde{p} + K^k(z^k - H^k\tilde{p}) \quad \ldots\ldots\ldots (2.9)$$

$$P^k = (I - K^k H^k)\bar{P} \quad \ldots\ldots\ldots (2.10)$$

$$K^k = \bar{P}H^{kT}(H^k\bar{P}H^{kT} + R)^{-1} \quad \ldots (2.11)$$

where $R = E[vv^T]$, diagonal; and to reiterate one takes $\tilde{p}^{k+1} = \hat{p}^k$ in (2.6) until one gets:

$$\hat{p}^{kc} = \tilde{p} \quad \ldots\ldots\ldots\ldots\ldots (2.12)$$

$$P^{kc} \cong E[(p - \hat{p})(p - \hat{p})^T] \ldots (2.13)$$

If one further wants to explore the possibilities given by this approach, one should notice that:

(i) since the components of the error v in (2.8) are uncorrelated (R diagonal), the recursive algorithm of (2.9) to (2.11) can be used to process z^k componentwise, thus avoiding the need of matrix inversion in (2.11);

(ii) if it happens that a new set of pairs is to be considered in the training of the neural net, one only has to take the values in (2.12) and (2.13) as a priori information ($\bar{p} \leftarrow \hat{p}$, $\bar{P} \leftarrow P^{kc}$) to process the new information;

(iii) there are many well developed results to guarantee a good performance to the estimator of (2.9) to (2.11), as for example the factorization methods (Bierman, 1977) to improve numerical behavior and the methods of noise adaptive estimation (Jazwinski, 1970; Rios Neto and Kuga, 1985) to guarantee both good numerical behavior and statistical consistency.

3. CONCLUSIONS

Motivated by the possibility of giving an interdisciplinary contribution, the author has used his experience in stochastic control and parameter estimation to formulate the problem of supervised training of neural nets as one of parameter stochastic estimation.

This approach leads to an algorithm which is more general and more realistic than the usual backpropagation. If a full version (as in (2.9) to (2.11)) is used one looses part of the paralel processing characteristic of the backpropagation algorithm. However, as shown in Chen and Billings (1992), for algorithms with the structure of a recursive least squares it is possible to get simplified versions still more general than backpropagation but wich preserve the paralel processing capability.

It is expected that the approach of parameter stochastic estimation as presented in its full version will be useful in off line neural nets training, and will provide simplified versions which will be equally useful in on line, real time schemes.

4. REFERENCES

BIERMAN, G.J. "Factorization Methods for Discrete Sequential Estimation", Academic Press, 1977.

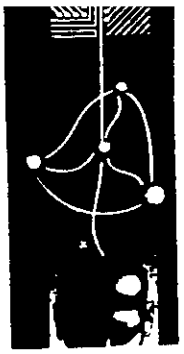
CHEN, S.; BILLINGS, S.A. "Neural Networks for Nonlinear Dynamic System Modelling and Identification", Int. J. Control, 56(2), 1992.

JAZWINSKI, A.H. "Stochastic Processes and Filtering Theory", Academic Press, N.Y., 1970.

ORLANDO, V.; RIOS NETO, A. "Aplicação de Técnica de Ruído Adaptativo em Estimação de Estado à Suavização de Dados", Procs. 1º Latin American Congress of Automatic and 5º Brazilian Congress of Automatica, Campina Grande, Paraíba, Brasil, 1984.

RIOS NETO, A.; CRUZ, J.J. "Design of a Stochastic Linear Discrete Regulator Based on an Optimal State Estimation Approach", SBA: Controle e Automação, 2(3), 1990.

RIOS NETO, A.; KUGA, H.K. "Kalman Filtering State Noise Adaptative Estimation", Procs. of Second IASTED Int. Conf. in Telecommunication and Control- TELECON'85, Rio de Janeiro, RJ, Brasil, 1985.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajuba
Itajuba, 24 a 27 de outubro de 1994

REDES NEURAIS COM DENSIDADE DE PROBABILIDADE UNIFORME NA ENTRADA-SAIDA

C.E.Pedreira and E.Parente
Dept. de Engenharia Elétrica
PUC-Rio C.P. 38063
22452-970 Rio de Janeiro
pedreira@ele.puc-rio.br

RESUMO: Neste artigo introduz-se um novo algorítimo de aprendizado que permite ter entradas as quais se atribui distribuições com densidade de probabilidade uniforme. A rede fornece na saida tambem uma distribuição com densidade de probabilidade uniforme. Este enfoque possui algumas vantagens relevantes com relação ao algorítimo de retropropagação original, sendo a principal destas a possibilidade de se operar com a falta eventual de alguma entrada sem a necessidade de um novo treinamento. Esta técnica é particularmente interessante em problemas com incertezas nas entradas e também quando se deseja estabelecer uma faixa de tolerância para a saída da rede. A função de custo proposta procura ao mesmo tempo incluir o intervalo de saída no intervalo alvo além de minimizar a distância entre seus centros. Simulações numéricas preliminares foram realizadas com o objetivo de ilustrar os resultados teóricos apresentados.

INTRODUÇÃO

O algoritimo de treinamento por retropropagação conforme originalmente proposto em [1] restringe sua aplicação a entrada com valores reais. Esta restrição leva a algumas limitações em uma classe importante de aplicações. O problema mais sério é o que se refere a ausência de parte dos atributos de entrada quando da fase de operação. Suponha-se por exemplo que após uma longa seção de treinamento em uma rede com n entradas, um destes dados não se encontra disponivel. Como proceder então? Retreinar uma nova arquitetura com n-1 entradas, normalmente não seria uma boa solução. Este problema de falta de atributos foi por algum tempo uma questão aberta do ponto vista teórico que possui grande relevância no que tange as aplicações. O uso de aritmética de intervalos [2] como ferramenta para solucionar esta questão foi originalmente proposto por Ishibuchi et al. em [3] dentro de um contexto de classificação de dois grupos, e de forma mais geral por Pedreira e Parente em [4]. O contexto exposto neste artigo propõe, no lugar de entradas compostas de números reais, entradas que assumem valores de intervalos reais. Deste modo, a rede estará sendo alimentada por intervalos reais fechados, ou equivalentemente, por distribuições com densidade de probabilidade uniforme. No caso exposto anteriormente, de não se ter acesso a algumas das entradas, se deverá alimentar a rede com o intervalo de máxima largura. Assim sendo, se estará informando à rede a máxima incerteza para o parâmetro em questão. Vale notar que qualquer número real pode ser representado como um intervalo real fechado e portanto, no caso de todas as entradas assumirem valores em $\Re$, o algoritimo aqui proposto se reduz ao esquema clássico de retropropagação. Diferentemente desta técnica tradicional e do algoritimo

proposto em [4], o alvo da rede é também um intervalo real fechado. Atraves desta modificação se torna possivel pré estabelecer a tolerância requerida pela aplicação em questão. Este enfoque parece ser bastante mais interessante do que, como é feito na tecnica tradicional, impor tolerância zero, o que muitas vezes não é realistico no que concerne a aplicações. Note-se que no enfoque aqui proposto é possivel controlar a precisão exigida em cada uma das variáveis de saída de modo independente. Questões ligadas a problemas com super treinamento e capacidade de generalização deste novo algoritimo permanecem em aberto e fogem ao escopo do presente artigo.

PRELIMINARES

Considere-se, por simplicidade de notação, uma rede do tipo *feedforward* com uma camada oculta e apenas uma unidade de saída. Assume-se que cada unidade da camada de entrada recebe, ao invés de um número real, um intervalo.

Notação

Considere-se a apresentação do padrão p. Sejam ε_{ip}^{L} e ε_{ip}^{U} , $i = 1,2,.....n$ os limites inferior e superior da i-ésima entrada respectivamente. o_{jp}^{L} e o_{jp}^{U} , $j = 1,2,....m$ denotam os limites inferior e superior da saída da j-ésima unidade da camada oculta, e O_p^L, O_p^U são os limites inferior e superior da unidade de saída da rede. Os n x m pesos de entrada são representados por w_{ji} , equanto que v_j são os m pesos que ligam a camada oculta à unidade de saída. A saída alvo, para cada padrão p tem como limites t_p^L e t_p^U. Seja $\overline{Y}$ o intervalo que tem como limites Y^L e Y^U.

Definições Básicas

Define-se a combinação linear dos intervalos de entrada, quando da apresentação do padrão p, como:

$$\overline{net}_{jp} \equiv \sum_{i=1}^{n} w_{ji}\bar{\varepsilon}_{ip} + \theta_j, \text{ para cada } j = 1,2,.....m \qquad (1)$$

enquanto que a combinação linear das saídas das unidades da camada oculta é dada por:

$$\overline{NET}_p \equiv \sum_{j=1}^{m} v_j\bar{o}_{jp} + \theta \qquad (2)$$

onde θ_j e θ são os termos polarizadores.
Nosso problema pode ser então estabelecido como: calcular os pesos da rede tal que a função de custo $E \equiv \sum_p e_p$ é minimizada. O custo devido ao padrão p é definido como:

$$e_p = \left(\frac{O_p^L+O_p^U}{2} - \frac{t_p^L+t_p^U}{2}\right)^2 + g(t_p^L - O_p^L) + g(O_p^U - t_p^U) \qquad (3)$$

onde,

$$g(x) = \begin{cases} 0, & x < 0 \\ x^2, & 0 \le x < \alpha \\ 2\alpha(x-\alpha)+\alpha^2, & x \ge \alpha \end{cases}$$

Esta função de custo reflete um problema de compromisso. Sua minimização leva à coincidência dos centros do intervalo de saída e do intervalo alvo ao mesmo tempo em que procura a inclusão do intervalo de saída no intervalo alvo. Este compromisso é balanceado pelo parâmetro α. O objetivo central é então definir um algoritimo tipo gradiente que modifique os pesos w_{ji} e v_j para i=1,2,...n , j=1,2,...m tal que a função de custo (3) seja minimizada.

Resultados Preliminares

Seja $I(\Re)$ o conjunto de todos os intervalos reais fechados. Note que qualquer número $x \in \Re$ pode ser considerado um elemento especial $[x,x]$ de $I(\Re)$. Sejam $\overline{X}, \overline{Y} \in I(\Re)$. As seguintes operações binárias podem ser calculadas como segue [2]:

$$\overline{X} + \overline{Y} = [X^L + Y^L, X^U + Y^U] \qquad (4)$$

$$\overline{X} - \overline{Y} = [X^L - Y^U, X^U - Y^L] \qquad (5)$$

$$\overline{X} \cdot \overline{Y} = [\min(X^L Y^L, X^L Y^U, X^U Y^L, X^U Y^U), \max(X^L Y^L, X^L Y^U, X^U Y^L, X^U Y^U)] \qquad (6)$$

$$\overline{X} \div \overline{Y} = [X^L, X^U] \cdot [1/Y^U, 1/Y^L] \qquad (7)$$

$$e^{\overline{X}} = [e^{X^L}, e^{X^U}] \qquad (8)$$

Assume-se que $0 \in \overline{Y}$ em caso de divisão (assumir $Y^U \neq 0$ e $Y^L \neq 0$ não e suficiente). Note-se que $I(\Re)$ é fechado sob as operações (4) - (8). Desta propriedade resulta que a aplicação do algoritimo proposto neste artigo implicará em que a saída da rede seja sempre um intervalo fechado. Além disso, pode-se provar [2] que se tem: (i) comutatividade e associatividade para adição e multiplicação; (ii) [0,0] e [1,1] são os únicos elementos neutros com relação a adição e multiplicação, respectivamente; e (iii) $I(\Re)$ não tem divisores zero.

Seja $\overline{Y} \in I(\Re)$ um intervalo pontual, i.e., $\overline{Y} = [y, y]$ onde $y \in \Re$. Tem-se então de (6) que:

$$\overline{X} \cdot \overline{Y} = [\min(Y^L y, X^U y), \max(Y^L y, X^U y)] \quad \forall \overline{X} \in I(\Re)$$

e portanto

$$\begin{aligned} y \cdot \overline{X} &= [yX^L, yX^U] \quad \text{para } y \geq 0 \quad \forall \overline{X} \in I(\Re) \\ y \cdot \overline{X} &= [yX^U, yX^L] \quad \text{para } y < 0 \quad \forall \overline{X} \in I(\Re) \end{aligned} \tag{9}$$

RESULTADOS PRINCIPAIS

Pode-se agora estabelecer um procedimento para calcular as mudanças nos pesos que minimizarão a contribuição do padrão p, e_p, na função de custo E. Vamos considerar a seguinte regra Delta:

$$\begin{aligned} \Delta_p v_j(s+1) &= \eta(-\partial e_p / \partial v_j) + \mu \Delta_p v_j(s) \\ \Delta_p w_{ji}(s+1) &= \eta(-\partial e_p / \partial w_{ji}) + \mu \Delta_p w_{ji}(s) \end{aligned} \tag{10}$$

onde η e μ representam os parâmetros de aprendizado e momento, respectivamente. A dificuldade restante está relacionada ao cálculo das derivadas parciais de e_p. De (1),(2) e (9) obtem-se:

$$\overline{net}_{jp} = \sum_{i=1}^{n} \left[w_{ji} \varepsilon_{ip}^{A}, w_{ji} \varepsilon_{ip}^{B} \right] + \theta_j \qquad \forall j = 1, 2, \ldots, m \ ,$$

e

$$\overline{NET}_p = \sum_{i=1}^{m} \left[v_i o_{ip}^{C}, v_i o_{ip}^{D} \right] - \theta$$

onde

A = L , B = U se $w_{ji} \geq 0$,
e A = U , B = L caso contrário,

C = L , D = U se $v_j \geq 0$,
e C = U , D = L caso contrário.

Com relação as unidades da camada oculta tem-se que:

$$\overline{o}_{jp} = f(\overline{net}_{jp})$$

e para a unidade de saída:

$$\overline{O}_p = f(\overline{NET}_p)$$

onde a função de ativação: $f: I(\Re) \to I(\Re)$ é uma logistica generalizada, i.e,

$$f(\overline{X}) = 1/(1 + \exp(-\overline{X})) \quad \forall \overline{X} \in I(\Re).$$

Note que, esta função logística generalizada pode ser definida por causa de (4),(7) e (8).

Define-se na sequência os seguintes parâmetros auxiliares:

$\Phi \equiv L$ se $w_{ji} \geq 0$, $\Phi \equiv U$ caso contrário

$\Psi \equiv U$ se $w_{ji} \geq 0$, $\Psi \equiv L$ caso contrário

$K \equiv 1$ se $v_j \geq 0$, $K \equiv 0$ caso contrário

Aplicando as operações básicas de aritimética de intervalos (4) - (7), e a regra da cadeia obtem-se:

(i) Para os pesos "entrada-oculta" :

$$\frac{\partial e_p}{\partial w_{ji}} =$$

$$\frac{\partial e_p}{\partial O_p^L} \frac{\partial O_p^L}{\partial NET_p^L} \left(\frac{\partial NET_p^L}{\partial o_{jp}^L} \frac{\partial o_{jp}^L}{\partial net_{jp}^L} \frac{\partial net_{jp}^L}{\partial w_{ji}} + \frac{\partial NET_p^L}{\partial o_{jp}^U} \frac{\partial o_{jp}^U}{\partial net_{jp}^U} \frac{\partial net_{jp}^U}{\partial w_{ji}} \right) +$$

$$\frac{\partial e_p}{\partial O_p^U} \frac{\partial O_p^U}{\partial NET_p^U} \left(\frac{\partial NET_p^U}{\partial o_{jp}^L} \frac{\partial o_{jp}^L}{\partial net_{jp}^L} \frac{\partial net_{jp}^L}{\partial w_{ji}} + \frac{\partial NET_p^U}{\partial o_{jp}^U} \frac{\partial o_{jp}^U}{\partial net_{jp}^U} \frac{\partial net_{jp}^U}{\partial w_{ji}} \right),$$

e então

$$\frac{\partial e_p}{\partial w_{ji}} = \left(\frac{\alpha}{2} \right) (2t_p - O_p^U - O_p^L)(\xi + \varpi) + \beta(\varpi - \xi) \tag{11}$$

onde

$$\xi \equiv K(1-O_p^L)O_p^L v_j o_{jp}^L (1-o_{jp}^L)\varepsilon_{ip}^{\Phi} +$$

$$+(1-K)(1-O_p^L)O_p^L v_j o_{jp}^U (1-o_{jp}^U)\varepsilon_{ip}^{\Psi}$$

$$\varpi \equiv (1-K)(1-O_p^U)O_p^U v_j o_{jp}^L (1-o_{jp}^L)\varepsilon_{ip}^{\Phi} +$$
$$+K(1-O_p^U)O_p^U v_j o_{jp}^U (1-o_{jp}^U)\varepsilon_{ip}^{\Psi})$$

(ii) Para os pesos "oculta - saída" :

$$\frac{\partial e_p}{\partial v_j} = \frac{\partial e_p}{\partial O_p^U}\frac{\partial O_p^U}{\partial NET_p^U}\frac{\partial NET_p^U}{\partial v_j} + \frac{\partial e_p}{\partial O_p^L}\frac{\partial O_p^L}{\partial NET_p^L}\frac{\partial NET_p^L}{\partial v_j}$$

$$= (\frac{\alpha}{\sqrt{\cdot}}(2t_p - O_p^U - O_p^L)(\pi+\sigma) + \beta(\sigma-\pi) \qquad (12)$$

onde

$$\pi \equiv o_{jp}^{\Psi} O_p^U (1-O_p^U) \quad \text{e} \quad \sigma \equiv o_{jp}^{\Phi} O_p^L (1-O_p^L)$$

Aplicando (11) e (12) em (10) pode-se treinar apropriadamente a Rede Neural de modo a minimizar a função de custo proposta.

RESULTADOS NUMÉRICOS PRELIMINARES

Os resultados numéricos aqui apresentados tem como objetivo ilustrar a técnica proposta, sem qualquer intenção de fazer uma análise detalhada do comportamento computacional desta metodologia ou de propor um novo sistema de diagnóstico inteligente. Simulou-se um sistema hipotético de apoio a decisão médica supondo-se que quatro exames são aplicados a cada um dos pacientes, sendo o primeiro destes mais relevante para a elaboração do diagnóstico do que os demais. A saída da rede indicará um índice que servirá de apoio a decisão médica. Assume-se que "0" e "1" indicam diagnóstico negativo e positivo respectivamente. Deste modo um indicador próximo a "1" apontará um paciente com boa saúde. Por outro lado, uma entrada igual a "1" reflete um exame cujo resultado foi positivo.

Em todos os exemplos que seguem a Rede Neural tem arquitetura (4:2:1). O seguinte conjunto de treinamento foi apresentado:

TABELA I - Padrões de treinamento

Apres.	Entrada				Alvo
	1	2	3	4	
1	1	1	1	1	1
2	1	1	1	0	1
3	1	1	0	0	1
4	1	0	0	0	0
5	1	1	0	1	1
6	1	0	1	0	1
7	1	0	1	1	1
8	1	0	0	1	1
9	0	0	0	0	0
10	0	1	0	0	0
11	0	1	1	0	0
12	0	1	1	1	1
13	0	0	1	1	0
14	0	0	0	1	0
15	0	1	0	1	0
16	0	0	1	0	0

Depois de uma sessão de treinamento de aproximadamente 4000 interações (Figura 1) obteve-se como saída correspondente às 16 apresentações de entrada os valores mostrados na tabela II.

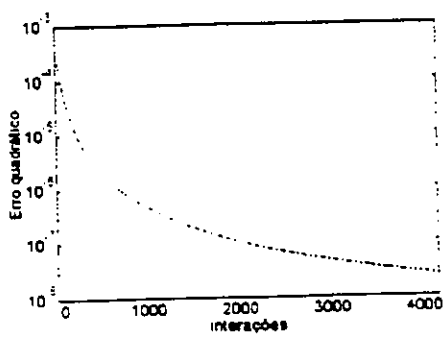


Figura 1

TABELA II - Saida da rede treinada

Apres.	Saida	Apres.	Saida
1	0,9998	9	0,0007
2	0,9996	10	0,0004
3	0,9876	11	0,0146
4	0,0190	12	0,9833
5	0,9996	13	0,0146
6	0,9996	14	0,0007
7	0,9876	15	0,0146
8	0,9876	16	0,0007

Note-se que foram encontradas somente saídas com valores reais, i.e. com limites superior e inferior dos intervalos iguais. Este resultado era esperado uma vez que todas as entradas apresentadas tambem foram pontuais. Em seguida o sistema foi colocado em operação para o caso do terceiro exame não ter sido realizado, i.e. entrada = (1, 1 , [0 , 1] , 0), e obteve-se como saída [0,9876 , 0,9996]. Este resultado parece coerente, uma vez que o exame não realizado não é o principal. Por outro lado, se omitirmos o primeiro exame, i.e. entrada = ([0 , 1] , 1 , 0 , 0), teria-se como saída [0,0007 , 0,9876], indicando, como esperado, uma saída indefinida. Na próxima ilustração usou-se o conjunto de treinamento exposto na tabela III, onde A = [0,8 , 1] e B = [0 , 0,2]. A rede foi treinada até atingir um somatório do erro médio quadrático de aproximadamente 10^{-7} . Esta rede foi operada posteriormente com entrada ([0 , 0,4] , 1 , 1 , 0) , obtendo-se como saída [0,9988 , 1,0000].

COMENTÁRIOS FINAIS

Neste artigo foram apresentados novos resultados teóricos que permitem treinar uma Rede Neural através de retropropagação do erro, com atributos que assumem valores de distribuições uniformes. Talvez a principal vantagem desta metodologia seja a possibilidade de se poder lidar com a eventual falta de alguns dos atributos de entrada sem a necessidade de um novo treinamento. Outro ponto que julgamos relevante é a formulação de modelo com tolerância pré determinada, para cada uma das saidas da rede. A intenção desta contribuição esta no plano teórico, embora visando fundamentalmente viabilizar uma classe relevante de aplicações, não se propõe a fazer estudo do comportamento numérico sistemático do algoritimo proposto, que fica assim para futuras investigações.

TABELA III - Apresentação de intervalos na entrada

Apres.	Entrada				Alvo
	1	2	3	4	
1	A	1	1	1	1
2	A	1	1	0	1
3	A	1	0	0	1
4	A	0	0	0	0
5	A	1	0	1	1
6	A	0	1	0	1
7	A	0	1	1	1
8	A	0	0	1	1
9	B	0	0	0	0
10	B	1	0	0	0
11	B	1	1	0	0
12	B	1	1	1	1
13	B	0	1	1	0
14	B	0	0	1	0
15	B	1	0	1	0
16	B	0	1	0	0

AGRADECIMENTOS

Os autores desejam agradecer a H.V. Carneiro e L.E. Sampaio pela cuidadosa leitura de revisão.

REFERÊNCIAS:

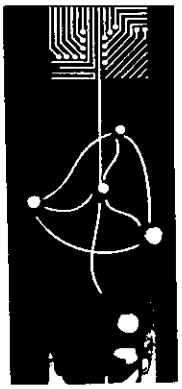
[1] D.E.Rumelhart, G.E.Hinton e R.J. Williams "Learning Internal Representation by Error Propagation" in Parallel Distributed

Processing, Vol 1, ed. Rumelhart e McClelland, 1986.

[2] G.Alefeld e J.Herzberger, "Introduction to Interval Computation" , Academic Press, N.Y.,1983.

[3] H.Ishibuchi, A.Miyazaki, K.Kwon e H.Tanaka, "Learning from Incomplete Training Data with Missing Values ansd Medical Application", Proc. of Int. Joint Conf. on Neural Networks, pp1871, Nagoya, Japan, Oct 1993.

[4] C.E.Pedreira e E.Parente, "An Interval Computation Approach To Backpropagation" Proc. of 1994 IEEE Workshop on Neural Networks For Signal Processing, Ermioni, Grécia, Set. de 1994.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

OPTIMAL ESTIMATE TRAINING 2 (OET-2)

Alexandre Pinto Alves da Silva[1]
Victor Hugo Quintana[2]
Germano Lambert Torres[1]

1 - ESCOLA FEDERAL DE ENGENHARIA DE ITAJUBÁ
2 - UNIVERSITY OF WATERLOO - CANADA

Abstract - A supervised learning algorithm for multilayer perceptrons (Optimal Estimate Training 2 - OET2) has been developed to overcome shortcomings of the generalized delta rule with backward error propagation. OET2 is particularly useful for real-world pattern recognition problems where very large training sets are necessary.

1. INTRODUCTION

The majority of the applications of ANNs have employed semi-linear feed-forward nets (multilayer perceptrons), using the generalized delta rule with backward error propagation for the learning process [1]. However, it is almost impossible to treat real-world problems utilizing this learning rule. The principal shortcomings of the generalized delta rule with backpropagation of error are as follows:

- it generally takes a long time to converge;
- it does not scale well (usually the training time grows exponentially as the number of neurons increase);
- it requires a choice of good parameters for the learning process (learning rate, constant of the momentum term and slope of the activation function); and
- as a steepest-descent method it could stop at a local minimum.

Despite many attempts to improve the performance of the generalized delta rule with backward error propagation [2, 3], and to find an alternative to it [4, 5], there is still no efficient and reliable method to train a multilayer perceptron. The main problem is the high nonlinearity of the error function.

The Functional Link Net (FLN) [6], a high-order net, has been tried with more success than a multilayer architecture. The basic idea is to avoid the necessity of hidden layers by performing a nonlinear transformation of the input pattern (enhancement of the original input pattern) before it is supplied to the input layer of the network. Without hidden layers, the learning process can be pursued using the delta rule instead of the generalized delta rule. Consequently, the FLN takes less time to converge.

Assuming that the input pattern enhancement is done properly, it has been shown that the FLN can also achieve more accurate results. However, besides the fact that the number of new inputs can increase considerably, depending on the number of original inputs and independent training patterns, it is not an easy task to define a general procedure that will guarantee the effectiveness and efficiency of the input pattern enhancement.

In order to overcome the problems mentioned above, a new method for loading information into a multilayer perceptron is proposed in this paper. The idea is to use successive quadratic approximations for the error function, until getting into the proximity of the global minimum.

2. A SUPERVISED LEARNING TECHNIQUE FOR VERY LARGE TRAINING SETS

Reference [7] presents a comparison study between Optimal Estimate Training (OET) and the generalized delta rule with backward error propagation. The task is to restore a data bit stream corrupted by a diffusive point-spread function. The results obtained are an indication that OET can be much faster and much more accurate than the other technique.

OET is a supervised learning technique for multilayer perceptrons. The sets of input and output patterns are represented in matrix form. An optimum nonlinear mapping that associates input training patterns to output training patterns is found by successively solving linear systems using direct methods, which

in fact is a procedure based on nonlinear least-squares methods (local approximation of the error function by a quadratic function, and the exact minimization of such an approximate function). The Moore-Penrose pseudoinverse is solved explicitly in different steps of the OET. An important feature of this technique is that the scaling problem is very well characterized. Based on the complexity of the operations involved ($O[r.m^2]$, where r is the number of input training patterns and m their dimension), it is easy to estimate how the OET learning rate will scale with network size.

3. OPTIMAL ESTIMATE TRAINING

The general idea of OET can be summarized using Figure 1. In Figure 1, $h_{in,j}$ (j = 0, 1, ..., m) is the set of input variables ($h_{in,0}$ represents an additional input with constant value equal to 1); w_{ji} is the set of interconnection weights leading from element j to element i, and $h_{out,i}$ represents the output of neuron i.

As usual, the propagation of the inputs is produced by a summation of the weighted inputs, i.e.,

$$z_{out,i} = \sum_{j=0}^{m} h_{in,j} w_{ji} \qquad (1)$$

The value of $h_{out,i}$ is calculated by the application of an activation function,

$$h_{out,i} = f(z_{out,i}) \qquad (2)$$

To perform the OET, a desired output value, $h_{d\text{-}out,i}$, is propagated back through the activation function and the corresponding desired intermediate result, $z_{d\text{-}out,i}$, is obtained. To execute this operation, it is necessary to employ a continuous invertible function such as the hyperbolic tangent. This function is suitable for the OET since the desired outputs can be positive or negative. Afterwards, the set of inputs and desired intermediate results can be interrelated to calculate optimal estimates for the set of interconnection weights, in a least-squares sense.

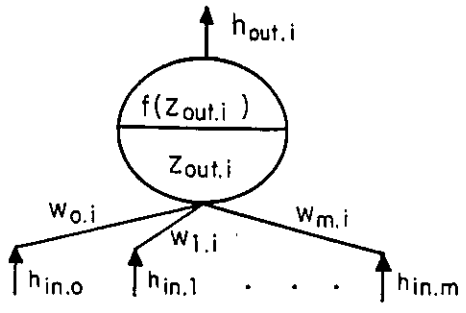


Figure 1: Representation of a neuron.

The procedure for training a network with one hidden layer (Figure 2) is now described. The extension of this training procedure for more than one hidden layer is straightforward.

The following notation is used in this paper:

$H_{d\text{-}out}$ → r by n desired output matrix;

H_{in} → r by m+1 input matrix;

$Z_{d\text{-}out}$ → $H_{d\text{-}out}$ propagated back through the output layer (r by n matrix); the i^{th} column of $Z_{d\text{-}out}$ is denoted by $\underline{z_{d\text{-}out}}^{i}$, and the j^{th} column of $Z_{d\text{-}out}^{t}$ is denoted by $\underline{z_{d\text{-}out}}^{j}$;

W_{out} → m+1 by n matrix of interconnection weights linking the hidden layer with the output layer; the i^{th} column of W_{out} is denoted by $\underline{w_{out}}^{i}$;

$H_{d\text{-}hid}$ → r by m+1 matrix; the j^{th} column of $H_{d\text{-}hid}^{t}$ is denoted by $h_{d\text{-}hid}^{j}$;

$Z_{d\text{-}hid}$ → $H_{d\text{-}hid}$ propagated back through the hidden layer (r by m+1 matrix); the k^{th} column of $Z_{d\text{-}hid}$ is denoted by $\underline{z_{d\text{-}hid}}^{k}$;

W_{hid} → m+1 by m+1 matrix of interconnection weights linking the input layer with the hidden layer; the k^{th} column of W_{hid} is denoted by $\underline{w_{hid}}^{k}$;

Z_{hid} → produced by multiplying H_{in} and W_{hid} (r by m+1 matrix);

H_{hid} → Z_{hid} propagated forward through the hidden layer (r by m+1 matrix);

Z_{out} → produced by multiplying H_{hid} and W_{out} (r by n matrix);

H_{out} → Z_{out} propagated forward through the output layer (r by n matrix);

r → number of input/output patterns in the training set;

m+1 → number of input neurons (which is the same as the number of neurons in the hidden layer); and

n → number of output neurons.

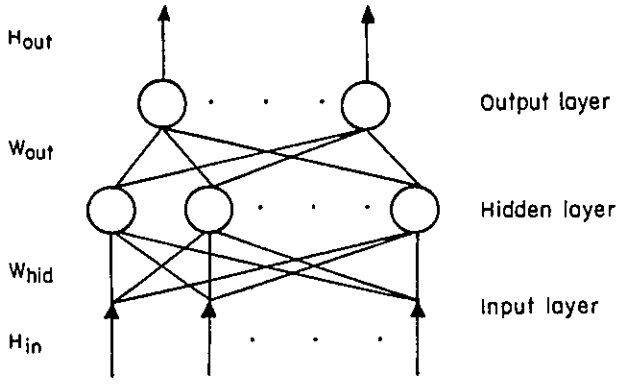


Figure 2: Fully connected feed-forward net with one hidden layer (input layer processing elements are data conduits)

The network training steps are [7]:

1) Calculate an initial value for W_{out} (this would be the final value for W_{out} if hidden layers were not used):

$$W_{out} = H_{in}^{-} Z_{d-out} \quad (3)$$

where

$$Z_{d-out} = \tanh^{-1}(H_{d-out}),$$

and

$$H_{in}^{-} = (H_{in}^{t} H_{in})^{-1} H_{in}^{t},$$

assuming that r is greater than m+1 (overdetermined system). The symbol + represents the generalized inverse or Moore-Penrose pseudoinverse.

2) Obtain H_{d-hid} that would produce Z_{d-out} given W_{out},

• if n = m+1,

$$H_{d-hid}^{t} = (W_{out}^{t})^{-1} Z_{d-out}^{t}, \quad (4)$$

assuming that W_{out}^{t} is non-singular;

• if n > m+1,

$$H_{d-hid}^{t} = (W_{out}^{t})^{+} Z_{d-out}^{t}, \quad (5)$$

$$(W_{out}^{t})^{-} = (W_{out} W_{out}^{t})^{-1} W_{out}, \quad (6)$$

overdetermined case;

and

• if n < m+1,

$$H_{d-hid}^{t} = (W_{out}^{t})^{+} Z_{d-out}^{t}, \quad (8)$$

$$(W_{out}^{t})^{-} = W_{out} (W_{out}^{t} W_{out})^{-1}, \quad (9)$$

underdetermined case.

3) Normalize the elements of H_{d-hid} to ensure that they will be situated within the output range of the hidden layer's activation functions,

$$H_{d-hid} = H_{d-hid} 0.99 / |\lambda|, \quad (10)$$

where λ is the element of H_{d-hid} with the largest magnitude. A multiplication factor, say 0.99, is used to obtain finite values for Z_{d-hid}, and to get advantage of the hyperbolic tangent nonlinearity.

4) Produce Z_{d-hid} using the inverse hyperbolic tangent,

$$Z_{d-hid} = \tanh^{-1}(H_{d-hid}) \quad (11)$$

5) Given H_{in} and Z_{d-hid}, calculate W_{hid},

$$W_{hid} = H_{in}^{-} Z_{d-hid} \quad (12)$$

6) Obtain the actual intermediate output for the hidden layer,

$$Z_{hid} = H_{in} W_{hid}, \quad (13)$$

and

$$H_{hid} = \tanh(Z_{hid}) \quad (14)$$

7) Recalculate W_{out},

$$W_{out} = H_{hid}^{-} Z_{d-out}, \quad (15)$$

where

$$H_{hid}^{-} = (H_{hid}^{t} H_{hid})^{-1} H_{hid}^{t}. \quad (16)$$

The network training is finished at step 7, i.e., once the matrices of interconnection weights W_{hid} and W_{out} are optimally calculated in the least-squares sense.

To obtain the ANN output for any set of inputs, the following steps are performed:

1) Combine H_{in} and W_{hid} to produce Z_{hid},

$$Z_{hid} = H_{in} W_{hid} \quad (17)$$

2) Treat Z_{hid} with the activation function (hyperbolic tangent),

$$H_{hid} = \tanh(Z_{hid}) \quad (18)$$

3) Combine H_{hid} and W_{out} to produce Z_{out},

$$Z_{out} = H_{hid} W_{out} \quad (19)$$

4) Treat Z_{out} with the activation function to generate H_{out} (the output of the network),

$$H_{out} = k \tanh(Z_{out}), \quad (20)$$

where k = 1.0101 (output normalization factor (1/0.99)).

4. OPTIMAL ESTIMATE TRAINING 2 (OET2)

In this section, the OET algorithm is extended in order to overcome its major drawbacks. Three new features are incorporated to the OET algorithm: numerical stability, input enhancement, and bounded interconnection weights.

4.1 Numerical Stability

Although the OET gives very promising results, some numerical problems with the matrix inversions of equations (4) and (16) are reported in [7]. These problems should have been expected because the conventional least-squares solution via normal equation is intrinsically prone to ill-conditioning problems. It can be shown that the condition number $c(H^t\ H)$ is the square of $c(H)$. This means that if H is not very well conditioned, $H^t\ H$ is ill-conditioned, and this may badly affect the final results, without any warning.

In order to improve the accuracy of the Shepanski's learning process, a QR (orthogonal) decomposition method could be used. There are several ways to define an orthogonal transformation. Since the patterns to be used in the training process of an ANN may be encountered sequentially, the Givens transformation is potentially suitable because it works by rows. With respect to this aspect, the Givens method is a better choice than the sequential algorithms based on the Kalman filter, which are also sensitive to numerical problems. An iterative method, such as the Preconditioned Conjugate Gradient (PCG), could be used to overcome ill-conditioning problems in the least-squares solution via a normal equation. However, the PCG method becomes more efficient than direct methods only when $H^t\ H$ has a sparse structure and a very large dimension (say, $> 10^3$), which is not usually the case since H_{in} is generally a dense matrix.

It is important to be aware of the cases in which H_{in} is not assumed to be sufficiently well conditioned. It could occur that during the computation, perturbations caused in H_{in} due to round-off errors replace H_{in} by a rank-deficient matrix. Notice that with OET there is a restriction related to the columns of the input matrix. If the values associated to a certain input channel can be obtained (approximately) as a linear combination of the corresponding values of other input channels, H_{in} will be rank-deficient ($H_{in}^t\ H_{in}$ singular). This arbitrary instability can be avoided using a stabilization method to detect the set of linearly independent columns [8]. A new input matrix can be formed by considering only the linearly independent columns (redundant information is discarded). A stabilization method shall also be applied in case of rank deficiency in W_{out}^t and/or H_{hid}, which must be treated as being of full rank.

4.2 Input Enhancement

A major limitation in the Shepanski's training technique is the necessity of having the same number of processing elements in the input and hidden layers. To overcome this shortcoming, an enhanced input representation scheme can be applied when it is necessary to increase the number of neurons in the hidden layer. Using the idea of facilitating the learning process by increasing the extent of nonlinearity, with a nonlinear transformation of the input pattern [6], the problem is solved in a better way than just including "dummy" input channels.

A strategy for input enhancement, similar to the one suggested in [9], is adopted in this work. The values of the most statistically correlated pairs of input channels, considering only the ones contained in the new full rank input matrix, are multiplied to form new entries. The appearance of a specific input in several products is not permitted. Limited diversity of information in noisy environments is not a safety policy. With this proposed technique, the necessary increase of the enhanced inputs is expected to be less than using the FLN because there is a corresponding increase in the number of hidden neurons.

4.3 Bounded Interconnection Weights

Interconnection weight bounds can be applied when it would be necessary to decrease the number of neurons in the hidden layer. Using OET, the number of neurons in the hidden layer cannot be decreased without reducing the dimensionality of the input patterns. An improvement on the ANN's generalization capability, produced by decreasing the number of hidden units (and, consequently, decreasing the number of interconnections), is expected to be obtained alternatively by reducing its interconnection weight bounds. This can be intuitively understood by noting that when weight bounds are applied, the total squared sum of errors can be controlled, allowing the training procedure to appropriately constrain the decision regions created by the network.

In fact, when an ANN is trained for classification or function approximation purposes, it is necessary to avoid the training data overfitting in order to obtain good generalization. This can be achieved by making a search for the simplest ANN (the one with less neurons and interconnections) which provides the minimum error rate for the testing set being used. When training with an iterative procedure like backpropagation, an alternative to avoid an exhaustive search for a good network architecture is to use cross-validation [10]. The idea is to interrupt the learning process as soon as the generalization for the test set begins to deteriorate.

"Easy learning" should also be avoided, i.e., when an input-output mapping can be established by assigning large weights (in comparative terms) to a small percentage of the total number of interconnections (e.g., when some features strongly characterize certain classes). Without noticing that "easy learning" is happening, one could be induced to reduce the ANN architecture. Low knowledge encoding redundancy caused by "easy learning" is highly undesirable in very noisy environments. Once more, interconnection weight bounds can be used to avoid "easy learning" and to get the benefits of knowledge encoding redundancy. A more uniform distribution of knowledge across the ANN is obtained by limiting the interconnection weight magnitudes.

The combination of input enhancement and bounded interconnection weights is not employed in this work because of the associated increase in the computational burden. However, this is a possibility that deserves investigation.

4.4 Proposed Training Algorithm

The proposed procedure for training a feed-forward network with one hidden layer is presented next. The encoding algorithm steps are as follows:

1) Determine an input matrix H_{in} (r X m),formed by m linearly independent input pattern features of the original H_{in} matrix.

2) Calculate an initial W_{out}:

$$H_{in} W_{out} = Z_{d-out} \tag{21}$$

$$\min_{\underline{w}_{out}^{i}} \left\| H_{in} \underline{w}_{out}^{i} - \underline{z}_{d-out}^{i} \right\|_2, \quad \text{for } i = 1, \ldots. \text{n} \tag{22}$$

assuming r greater than m.

3) Given W_{out}, obtain $H_{d\text{-}hid}$ that produces $Z_{d\text{-}out}$:

$$W_{out}^{t} H_{d-out}^{t} = Z_{d-out}^{t} \tag{23}$$

• if n ≥ m then:

$$\min_{\underline{h}_{d-hid}^{j}} \left\| W_{out}^{t} \underline{h}_{d-hid}^{j\,t} - \underline{z}_{d-out}^{j\,t} \right\|_2, \text{ for } j = 1, \ldots, \text{r} \tag{24}$$

• if n < m then:

$$\min \left\| \underline{h}_{d-hid}^{j\,t} \right\|_2, \text{ s.t., } W_{out}^{t} \underline{h}_{d-hid}^{j\,t} = \underline{z}_{d-out}^{j\,t}, \quad \text{for } j = 1, \ldots, \text{r} \tag{25}$$

4) Normalize the elements of $H_{d\text{-}hid}$ to assure that they will be located within the output range of the hidden layer's activation functions (hyperbolic tangent),

$$H_{d-hid} = H_{d-hid}\, 0.99 / |\lambda| \tag{26}$$

where λ is the element of $H_{d\text{-}hid}$ with the largest magnitude.

5) Calculate $Z_{d\text{-}hid}$ using the inverse hyperbolic tangent,

$$Z_{d-hid} = \tanh^{-1}(H_{d-hid}) \tag{27}$$

6) Given H_{in} and $Z_{d\text{-}hid}$, calculate W_{hid},

$$H_{in} W_{hid} = Z_{d-hid} \tag{28}$$

$$\min_{\underline{w}_{hid}^{k}} \left\| H_{in} \underline{w}_{hid}^{k} - \underline{z}_{d-hid}^{k} \right\|_2, \text{ s.t., } \left| w_{hid,l}^{k} \right| \le Bound1, \quad \text{for } k = 1, \ldots, m \text{ and } l = 1, \ldots, m, \tag{29}$$

where $w_{hid,l}^{k}$ is a compoment of $\underline{w}_{hid}^{k}$.

7) Obtain the actual intermediate output for the hidden layer,

$$Z_{hid} = H_{in} W_{hid}, \tag{30}$$

and

$$H_{hid} = \tanh(Z_{hid}) \tag{31}$$

8) Recalculate W_{out}:

$$H_{hid} W_{out} = Z_{d-out} \tag{32}$$

$$\min_{\underline{w}_{out}^{i}} \left\| H_{hid} \underline{w}_{out}^{i} - \underline{z}_{d-out}^{i} \right\|_2, \text{ s.t., } \left| w_{out,l}^{i} \right| \le Bound2, \quad \text{for } i = 1, \ldots, \text{n and } l = 1, \ldots, m. \tag{33}$$

where $w_{out,l}^{i}$ is a component of $\underline{w}_{out}^{i}$.

4.5 Finding Bounds

To find a good pair of bounds the following heuristic procedure is employed:

1) Train the ANN (one hidden layer) with no bounds. This allows maximum network capacity for representing nonlinear mappings.

2) Keep Bound2 set to a very large number. Set Bound1 approximately to the maximum value that produces a solution for W_{hid} which has all elements equal to this value.

3) Improve the ANN capacity of producing nonlinear mappings by gradually raising Bound1, while leaving Bound2 set to a very large number.

a) If this procedure immediately increases the error rate for the test set, then try an ANN without hidden layers. Otherwise, go to b).

b) Continue to raise Bound1 until the error rate for the test set starts to increase. Then, go to step 4. If the ANN generalization ability keeps improving until returning to the situation described in step 1, then go to c).

c) Apply an input enhancement scheme to increase the extent of nonlinearity allowed by the ANN.

4) Keep the best Bound1 value, and lower Bound2 to check if the capacity for representing nonlinearities is greater than necessary. Lower Bound2 until the error rate starts to increase again. The training process is finished.

Numerically stable methods for solving the linear least-squares problems (22), (24), (25), (29) and (33) are described in [8].

CONCLUSIONS

This paper has presented a learning algorithm, Optimal Estimate Training 2 (OET2), for multilayer perceptrons. OET2 is very appropriate for large training sets. Its execution time varies linearly with the number of patterns and quadratically with the dimension of the input patterns. OET2 does not require any choice of learning parameters. OET2 has proved to be orders of magnitude faster than the backpropagation algorithm in practical applications [11-14].

The following topics deserve further research effort:

- Inclusion of adaptation capacity in the OET2 algorithm. Row weighting in the orthogonal decomposition of the input matrix (H_{in}) can be used. The efect of past data can be reduced by gradually increasing the weighting as new data are acquired.

- Application of column weighting in the orthogonal decomposition of the input matrix (H_{in}) during the ANN training. The inverses of the variances of the input variables can be used as weighting factors. This procedure can improve the ANN generalization ability.

REFERENCES

[1] D.E. Rumelhart, G.E. Hinton, and R.J. Williams: "Learning internal representations by error propagation in parallel distributed processing", in *Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, MIT Press, 1986, pp. 318-362.

[2] L.-W. Chan and F. Fallside: "An adaptive training algorithm for back propagation networks", *Computer Speech and Language*, Vol. 2, 1987, pp. 205-218.

[3] R.A. Jacobs: "Increased rates of convergence through learning rate adaptation", *Neural Networks*, Vol. 1, 1988, pp. 295-307.

[4] S.E. Fahlman and C. Lebiere: "The cascade-correlation learning architecture", in *Advances in Neural Information Processing Systems 2*, D.S. Touretzky (Ed.), Morgan Kaufmann, 1990, pp. 524-532.

[5] W.T. Miller III, F.H. Glanz, and L.G. Kraft III: "CMAC: an associative neural network alternative to backpropagation", *Proceedings of the IEEE*, Vol. 78, Oct. 1990, pp. 1561-1567.

[6] Y.-H. Pao: *Adaptive Pattern Recognition and Neural Networks*, Addison Wesley, 1989.

[7] J.F. Shepanski: "Fast learning in artificial neural systems: multilayer perceptron training using optimal estimation", *Proc. IEEE 2nd Intern. Conf. Neural Nets*, San Diego, Jul. 1988, Vol. 1, pp. 465-472.

[8] C.L. Lawson and R.J. Hanson: *Solving Least Squares Problems*, Prentice-Hall, 1974.

[9] Y.-H. Pao and D.J. Sobajic: "Autonomous feature discovery for critical clearing time assessment", *Proc. 1st Symp. Expert Syst. Applications Power Syst.*, Stockholm-Helsinki, Aug. 1988.

[10] G.E. Hinton: "Using cross-validation to avoid overfitting", presented at *Neural Network Workshop*, Niagara-on-the-Lake, May 1990.

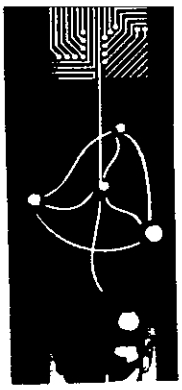
[11] A.P. Alves da Silva, V.H. Quintana, and G.K.H. Pang: "Solving data acquisition and processing problems in power systems using a pattern analysis approach", *IEE Proceedings Part C: Generation, Transmission and Distribution*, Vol. 138, July 1991, pp. 365-376.

[12] A.P. Alves da Silva, V.H. Quintana, and G.K.H. Pang: "Neural networks for topology determination of power systems", *Proc. 1st International Forum on Applications of Neural Networks to Power Systems* (IEEE Catalog Number: 91TH0374-9), Seattle, July 1991, pp. 297-301.

[13] A.P. Alves da Silva, A.M. Leite da Silva, J.C.S. de Souza, and M.B. Do Coutto Filho: "State forecasting based on artificial neural networks", *Proc. 11th Power Systems Computation Conference*, Avignon, August 1993, pp. 461-467.

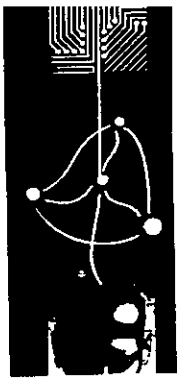
[14] L.E. Borges da Silva, A.P. Alves da Silva, G. Lambert Torres, and G. Roy: "An alternative neural network training algorithm for rreal time control", *Proc. IEEE International Symposium on Industrial Electronics*, Santiago, May 1994, pp. 72-76.

Anotações



HEURÍSTICAS

Anotações



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

CLASSIFICADORES NEURAIS USANDO BACKPROPAGATION COM ATENÇÃO SELETIVA

Marcelo C. Bossan[1], Luiz P. Calôba[2] e Jurandir Nadal[1]

[1]COPPE:Programa de Engenharia Biomédica e [2]COPPE/EE, Universidade Federal do Rio de Janeiro, Brazil

RESUMO

Uma das aplicações mais freqüentes das redes neurais é a classificação de padrões. Para tal, geralmente utiliza-se a estrutura *feedforward* com treinamento *backpropagation*, minimizando o erro médio quadrático (EMS) das saídas da rede. Este critério, razoável para a aproximação de funções, possui características que dificultam a obtenção da menor taxa possível de erros de classificação. A modificação no algoritmo *backpropagation* apresentada neste trabalho permite um melhor desempenho da rede neural em casos onde o espaço de padrões possui regiões pouco populosas juntamente com *clusters* com grande concentração de padrões, dirigindo a atenção do treinamento prioritariamente para os padrões ainda não aprendidos.

1.INTRODUÇÃO.

O algoritmo *backpropagation* geralmente é a primeira opção quando se deseja treinar uma rede neural *feedforward* multi-camadas. Sua característica de busca de descida segundo o gradiente, visando encontrar um ponto de mínimo na superfície de erro médio quadrático (EMQ) da saída, é muito desejável, na maioria dos casos, pois dá maior importância às entradas mais freqüentes e especial atenção às entradas mais deslocadas do centro da distribuição. Estas características tornam o algoritmo indicado para a aproximação de funções, onde um critério de mínimos quadrados é bastante razoável (Hecht-Nielsen, 1990).

Embora o problema da classificação de padrões possa ser entendido como o de encontrar uma função que associe padrões de entrada à uma classe, entre um número finito de classes (Lippman,1989), passos intermediários no sentido de reduzir o EMQ para o conjunto de padrões utilizado no treinamento da rede nem sempre implicam em redução instantânea do número de erros de classificação, podendo até mesmo causar um aumento no total de erros em algum instante durante o treinamento. Este algoritmo é reconhecidamente lento, sendo que muitos trabalhos vêm sendo publicados no sentido de reduzir o número de iterações

necessárias para a obtenção de um valor de EMQ satisfatório para o conjunto de treinamento e para o conjunto de avaliação (ver, p. ex., Hertz, Krogh e Palmer, 1990 e, Liguni et alii, 1992).

Uma alternativa possível para o treinamento de redes neurais *feedforward*, quando utilizadas para a classificação de padrões, é a modificação do algoritmo *backpropagation* com o objetivo de torna-lo mais sensível à quantidade de erros de classificação, durante os passos do treinamento da rede, ainda que em detrimento do EMQ para o conjunto de treinamento.

2.MODIFICAÇÃO NO ALGORITMO.

As redes neurais do tipo *feedforward* são as mais utilizadas atualmente para a classificação de padrões. Os padrões são apresentados à entrada desta rede que deve possuir uma saída correspondente à cada classe do espaço de padrões. As células utilizadas como exemplo terão funções de ativação sigmoidais. O objetivo do treinamento é fazer com que a rede tenha apenas uma de suas saídas ativada (igual a 1) para cada entrada aplicada e que esta saída corresponda à classe a que esta entrada pertença, ficando as demais saídas desativadas (iguais a 0). Como este comportamento ideal não é alcançável com um número finito de treinos, estabelece-se um limiar acima do qual uma saida é considerada ativada e abaixo do qual considerada desativada. Para células com saídas sigmoidais variando entre 0 e 1 utiliza-se normalmente o valor 0.5 como limiar. Quando estes valores são obtidos de forma correta para um determinado padrão, diz-se que a rede "aprendeu" a classificar este padrão.

O algoritmo normalmente aplicado para o treinamento da rede é o *backpropagation*, que utiliza um método de descida de gradiente para reduzir o erro médio quadrático para todo o conjunto de treinamento, expresso por

$$E[w] = \frac{1}{2}\sum_{u,i}\left[\zeta_i^u - O_i^u\right]^2$$

onde i e u representam respectivamente cada saída da rede e cada padrão aplicado à entrada da rede, ζ_i^u é a saída desejada (0 ou 1) e O_i^u a saída obtida .

O EMQ é fortemente dependente da freqüência de ocorrência de cada padrão. Qualquer tentativa de aproximar a O_i^u de um padrão muito freqüente de sua correspondente ζ_i^u, causará uma redução maior no EMQ, para todas as entradas, do que ocorreria com um padrão pouco freqüente. Como o algoritmo *backpropagation* sempre procura a redução mais rápida possível do EMQ, a partir da sua posição instantânea na curva, a rede tenderá a reduzir primeiro os erros devidos aos padrões mais freqüentes, até que estes erros sejam tão pequenos que os erros relacionados aos outros padrões se tornem significativos em relação aos erros dos padrões mais freqüentes. Só então os padrões menos freqüentes começarão a ser aprendidos.

Para um padrão já aprendido pela rede, qualquer tentativa de elevar a saída alta em direção ao seu valor ideal, 1, ou de reduzir as saídas baixas em direção à zero não contribuirá para a redução do número de erros da rede, mas este "reforço" será efetivo para a redução do EMQ da rede. Se este padrão for muito freqüente, em relação aos demais padrões do conjunto de treinamento, a rede se deterá na tentativa de realizar esta aproximação de valores, visando a redução do seu EMQ. Desta forma, a rede ficará por um longo número de seções de treinamento "presa" na tarefa de reduzir seu EMQ sem que isto cause qualquer redução no número de erros de classificação. Mais grave ainda, alguns padrões pouco freqüentes já aprendidos poderão até mesmo passar a ser classificados erroneamente durante esta aproximação, sendo necessário um grande número de seções de treinamento para que estes padrões possam voltar a ser classificados corretamente, o que pode resultar em um treinamento excessivamente lento.

Uma solução para o problema é tornar o algoritmo de treinamento da rede mais sensível às saídas correspondentes aos padrões erroneamente classificados e menos sensível aos erros das saídas correspondentes aos padrões já aprendidos. Definindo-se

$$e_i^u = \zeta_i^u - O_i^u$$

deseja-se que o algoritmo dê ênfase à redução dos erros onde $|e_i^u|>0.5$ e atenue os efeitos de valores de $|e_i^u|<0.5$ na busca do gradiente. Isto já é feito, de forma branda no algoritmo original, onde o erro "retropropagado" através das células da camada de saída é dado por

$$\delta_i^u = g'(h_i^u)\left[\zeta_i^u - O_i^u\right] = g'(h_i^u)\cdot f(e_i^u)$$

onde $g'(h_i^u)$ é a derivada da função de ativação da célula i da camada de saída quando o padrão u é aplicado à rede e $f(e_i^u)=e_i^u$. Neste caso, os valores de $f(e_i^u)=e_i^u$ para $|e_i^u|<0.5$, embora menores que para $|e_i^u|>0.5$, ainda permitem que a rede continue sendo treinada diminuindo o erro das entradas já aprendidas. Uma forma de se efetuar treinamento apenas quando ocorrer um erro de classificação é utilizar uma função $f(e_i^u)$ que seja não nula apenas para $|e_i^u|>0.5$, como a apresentada na Figura 1. O fato desta função não ser diferenciável impede seu uso em um algoritmo de gradiente, como o backpropagation. Para eliminar este problema, a solução desenvolvida foi a "suavização" da função

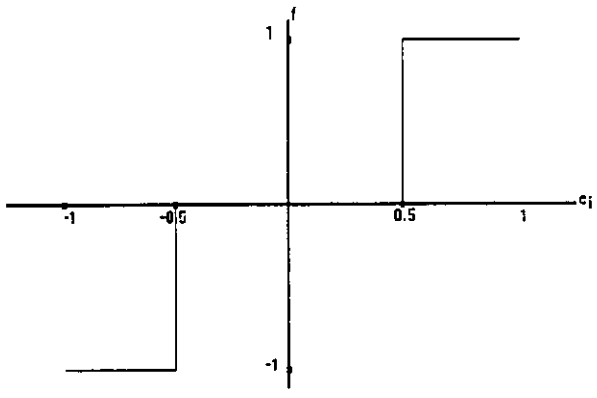


Figura 1 - $f(e_i)$ ideal para classificação de padrões.

através de uma aproximação sigmoidal, que pode ser obtida pela expressão

$$f(e_i^u) = \left(\frac{1}{1+e^{-\alpha(|e_i^u|-0.5)}} - a \right) \cdot b \cdot \text{sign}(e_i^u),$$

onde

$$a = \frac{1}{1+e^{\alpha/2}} \text{ , e } b = \frac{2+e^{\alpha/2}+e^{-\alpha/2}}{e^{\alpha/2}-e^{-\alpha/2}} .$$

Esta função tem como característica principal variar desde uma reta (como a do algoritmo original), para α próximo de 1, até a função idealizada da Figura 1 (semelhante à utilizada no treinamento do perceptron de Rosenblatt (1962)), para valores elevados de α. Para valores intermediários de α a função adquire um aspecto sigmoidal em cada um dos semi-planos. A Figura 2 apresenta o gráfico desta função para valores de α iguais a 1, 10 e 100.

Desta forma, aumentando-se α, $f(e_i^u)$ vai aos poucos reduzindo progressivamente seus valores para $|e_i^u|<0.5$ e aumentando seus valores para $|e_i^u|>0.5$.

A função custo F[w] associada ao algoritmo modificado tem mínimo em e_i^u igual a zero e é dada pela expressão

$$F[W] = \int f(x)dx$$
$$= \frac{2+e^{\alpha/2}+e^{-\alpha/2}}{e^{\alpha/2}-e^{-\alpha/2}} \cdot \sum_{iu} \left(\frac{1}{\alpha} \ln\left(1+e^{\alpha(|e_i^u|-0.5)}\right) - \frac{|e_i^u|}{1+e^{\alpha/2}} \right)$$

Com esta modificação o algoritmo torna-se capaz de reduzir a longa permanência do treinamento da rede na tentativa de

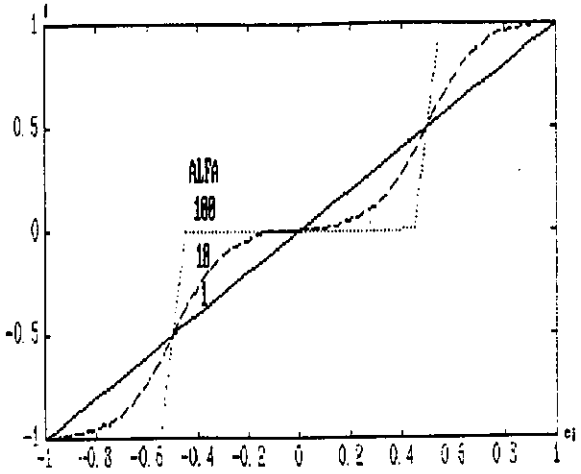


Figura 2 - Gráfico do $f(e_i)$ proposto para a modificação no algoritmo *backpropagation*.

aproximar de 0 ou 1 as saídas correspondentes aos padrões já aprendidos, liberando-a para o aprendizado de novos padrões. A redução no número de seções de treinamento aumenta com α, assim como também são aumentadas as regiões muito abruptas e muito planas da superfície de F[w] para o problema, elevando a possibilidade de paralisação da rede, como ocorre com os perceptrons de Rosenblatt (1962).

4.RESULTADOS

Dois diferentes conjuntos de padrões foram utilizados para a verificação do desempenho de redes neurais na classificação de padrões quando da introdução da alteração no algoritmo backpropagation proposta neste trabalho. No primeiro teste, foram considerados dois grupos de padrões de classes distintas, representados por cruzes e círculos cheios, cada um distribuído por uma região retangular composta por 1000 padrões. Acima da região à esquerda e

abaixo da região à esquerda, foi colocado 1 padrão de classe oposta à da região mais próxima (Figura 3). Treinando-se um único neurônio para a separação dos padrões deste espaço, utilizando-se o valor 0.1 para o passo de treinamento e 0.7 para o fator de momento, foram necessárias 200 apresentações de todo o conjunto de treinamento até que a rede fosse capaz de realizar corretamente o treinamento, obtendo resultado semelhante ao apresentado na Figura 3, onde a linha representa o separador gerado pelos pesos do neurônio. Utilizando-se a modificação proposta neste trabalho, com α igual a 100, foi possível a separação apresentada na Figura 3 com apenas 23 apresentações do conjunto de treinamento.

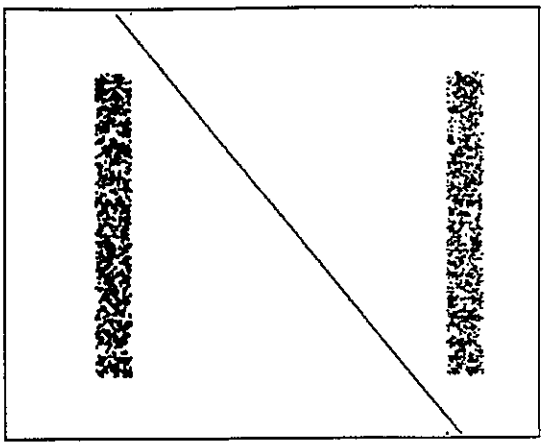


Figura 3 - Classificação do espaço de padrões do primeiro teste utilizando a modificação proposta neste trabalho.

Um segundo teste foi realizado utilizando uma rede de duas camadas (uma camada escondida), onde as vantagens da modificação proposta são ainda mais aparentes. Para tal, uma rede com 2 neurônios na primeira camada e 1 neurônio na segunda camada foi treinada com o algoritmo backpropagation para separar os padrões de um espaço formado por um padrão separados de outros 1000 de mesma classe (círculos cheios) por 1000 outros padrões de classe diferente (cruzes), conforme apresentado na Figura 4. Foram utilizados os valores 0.4 e 0.7 para, respectivamente, passo de treinamento e fator de momento. Mesmo após 10000 apresentações de todo o conjunto de treinamento, a rede não conseguiu classificar corretamente os padrões, separando apenas os dois grande blocos de dados. Utilizando-se a modificação proposta, com α igual a 100, foi possível a separação apresentada na Figura 4 com apenas 600 apresentações do conjunto de treinamento.

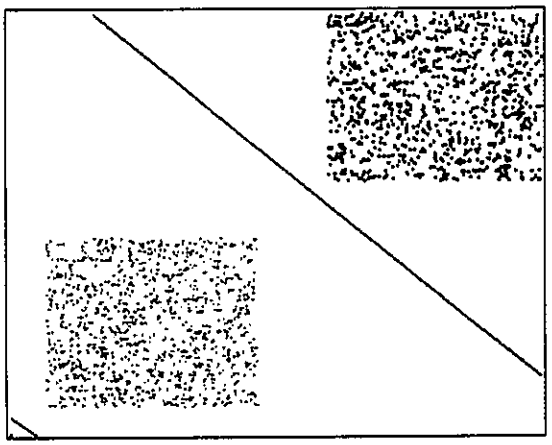


Figura 4 - Classificação do espaço de padrões do segundo teste utilizando a modificação proposta neste trabalho.

5.CONCLUSÕES.

Os testes apresentados mostram que a modificação proposta é capaz de resultar em aceleração do treinamento em casos onde existam padrões afastados de seus *clusters* principais, onde suas inclusões no grupo de padrões corretamente aprendidos pela rede não implicam em redução significativa de seu erro médio quadrático. Estes testes foram realizados em situações extremas, tanto no que concerne ao espaço de padrões quanto ao valor de α utilizado. Em casos reais, valores menores de α devem ser utilizados, podendo variar de 1, para um treinamento praticamente igual ao que se obtém ao utilizar o algoritmo backpropagation original, até valores tão elevados quanto 100, onde a rede comporta-se praticamente como um perceptron de Rosenblatt, com os seus conhecidos problemas de convergência.

O estudo deste possível novo algoritmo encontra-se ainda em fase inicial. Deve ainda ser definido um critério para o ajuste do valor de α em diferentes situações, ou de uma metodologia para a variação automática de α durante o treinamento. Uma alternativa aparentemente promissora é iniciar o treinamento com α igual a 1, permitindo um treinamento inicial praticamente idêntico ao do backpropagation original, elevando o valor de α quando for percebida uma redução significativa da velocidade de aprendizagem da rede. Porém, certamente o critério utilizado para a adaptação de α dependerá das características específicas do problema.

6.REFERÊNCIAS BIBLIOGRÁFICAS.

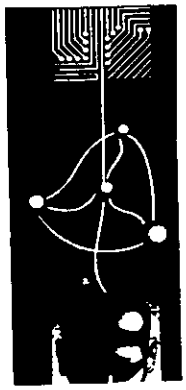
Hecht-Nielsen, R. (1990), *Neurocomputing*. Addison-Wesley Publishing Company, New York.

Hertz, J., Krogh, A. e Palmer, R.G. (1990), *Introduction to the Theory of Neural Computation*, Addison-Wesley Publishing Company,New York.

Liguni, Y., Sakai, H., e Tokumaru,H. (1992), "A real Time Learning Algorithm for a Multilayered Neural Network Based on the Extended Kalman Filter", *IEEE Transactions on Signal Processing*, Volume 40, Páginas 959-966, Abril,1992.

Lippman, R.P. (1989), "Pattern Classification Using Neural Networks", *IEEE Comunications Magazine*, páginas 47-64, Novembro.1989.

Rosenblatt, F. (1962), *Principles of Neurodinamics*, Spartan, NewYork.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

UM ACELERADOR GENÉTICO PARA REDES NEURONAIS

Antonius H. M. de Knegt
Mestrado em Eng. Elétrica
CPGEE - UFMG
D.E.T. / USIMINAS

Evandro de O. Araújo
Departamento de Eletrônica
Escola de Engenharia da UFMG
e-mail: earaujo@brufmg.bitnet

Gutemberg de S. Medeiros
Estudante de Eng. Elétrica
UFMG

Resumo

Este artigo apresenta um estudo sobre a utilização de Redes Neuronais em Controle de Processos analisando a viabilidade do emprego dos algoritmos genéticos para acelerar a convergência das redes. O princípio de funcionamento dos algoritmos genéticos é descrito. Resultados da utilização destes algoritmos com uma rede backpropagation para o cálculo da abertura entre os cilindros de um laminador são apresentados. A redução obtida no tempo de processamento desta aplicação nos indica um caminho bastante promissor a ser explorado.

1 Introdução

Estudos de sistemas computacionais inspirados em modelos biológicos remontam aos anos 40 (McCulloch, 1943; Hebb, 1949). Recentemente, a partir dos anos 80, o interesse por estes sistemas conexionistas vem crescendo de maneira bastante sólida, fruto de uma melhor compreensão de suas potencialidades e limitações. O surgimento de novas estruturas para estas redes (HOPFIELD, 1982; LIPMANN, 1987), avanços significativos nos algoritmos de aprendizado e a disponibilidade de processadores cada vez mais rápidos são os fatores que mais contribuiram para a popularização das redes neuronais artificiais. A computação neuronal representa uma maneira efetiva de se tentar incorporar na máquina algumas das potencialidades do cérebro humano tais como aprendizagem com a experiência, raciocínio de senso comum, a capacidade de generalizar e tomar decisões num ambiente de conhecimento incompleto, incerto e impreciso. Uma rede neuronal do tipo "backpropagation" pode ser usada em conjunto com um controlador nebuloso para reduzir os tempos de processamento e atenuar os problemas de memória computacional necessária à aplicação. Uma vez treinada a rede, é muito mais rápido obter os resultados com ela do que consultar uma "look-up table" que pode ser muito grande se o nível de quantização for grande e/ou a partição nebulosa fina (LEE, 1990). As redes neuronais têm também um aspecto de adaptabilidade inerente muito forte, tornando

bastante atraente a utilização destas em sistemas adaptativos. A rede, contudo, padece de alguns inconvenientes. O principal, a nosso ver, é a insegurança quanto à convergência. Os algoritmos genéticos são algoritmos de procura inspirados em modelos genéticos e têm sido pesquisados desde os anos 70 (Holland, 1975). Eles executam uma busca focalizada no espaço de soluções do problema através da acumulação de conhecimento durante a procura, conseguindo assim rapidez e economia de espaço de memória. Eles podem ser aplicados a uma ampla gama de problemas de otimização como, por exemplo, ao problema de seqüenciamento ótimo que se torna rapidamente intratável quando a dimensão cresce. A idéia de integração de um módulo genético com uma rede neuronal aparece como um caminho natural para tentar combinar as boas características destas duas técnicas, como será explicado na seção 4.

2 Redes Neuronais Artificiais

Uma rede neuronal artificial tal como um sistema nebuloso pode ser vista como estimadores sem modelo (KOSKO, 1991). Neste sentido, são aproximadores universais de funções bastante gerais, capazes de mapear vetores de entrada X em vetores de saída Y sem que seja necessária a utilização de um modelo matemático. Uma unidade básica de uma rede neuronal é um neurônio. A figura 1 mostra um neurônio constituído de N entradas ao qual foi acrescentado uma entrada adicional X_0 (polarização), para maior flexibilidade da rede, de valor 1.

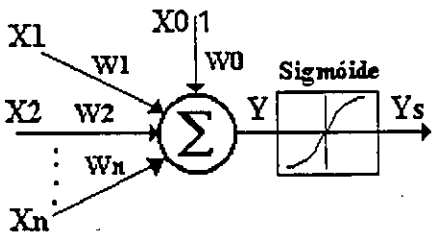


Figura 1

Na figura acima, $[W_0 \ldots W_N]$ é o vetor de pesos. Representando por **sgm(.)** uma função de ativação do neurônio, a operação realizada pelo neurônio sobre um vetor de entradas pode ser expressa por:

$$y_s = sgm\left(\sum_{k=0}^{N} X_k W_k\right)$$

Optamos, neste trabalho, por uma função de ativação monotônica crescente e derivável em qualquer ponto da forma:

$$y_s = \frac{1}{1+e^{-y}}$$

Os neurônios podem ser interligados, as saídas de uns compondo as entradas de outros, formando diversos níveis ou camadas. Utilizamos, aqui, estruturas neuronais não realimentadas ("feedforward") gerando por isto sempre sistemas estáveis.

2.1 Treinamento das redes neuronais

Treinar uma rede neuronal é encontrar um vetor **W** de pesos que possibilite a rede emular uma função qualquer, com uma precisão estipulada. A rede é treinada utilizando-se subconjuntos de dados, extraídos judiciosamente dos universos de entrada e saída. Após o treinamento, a rede é capaz de "generalizar", isto é, capaz de gerar valores aproximadamente corretos para um vetor de entradas arbitrário que não faz parte do subconjunto de treinamento. As redes neuronais podem ser treinadas através de um processo *supervisionado* ou *não supervisionado*. No primeiro método, os valores de saída da rede para cada apresentação das entradas são comparados com a correspondente saída desejada, gerando um vetor de erro que é então utilizado como orientador do processo de treinamento dos pesos. No caso do treinamento não supervisionado as entradas são agrupadas em classes, usualmente por similaridade ou

proximidade geométrica dos vetores de entrada. Os padrões podem ser definidos antes do início do treinamento ou a própria rede pode criar estas classes durante o processo de classificação.

2.2 A rede backpropagation

Trabalhamos com uma rede "feedforward" de 3 camadas (uma camada oculta) usando o modo de treinamento supervisionado. O esquema de aprendizado, chamado de "error backpropagation", baseia-se na minimização do gradiente do erro quadrático entre a saída desejada e a saída calculada pela rede. O treinamento no método backpropagation começa com a apresentação do vetor de entradas **X** à camada de entrada da rede. Este vetor é propagado através da rede gerando o vetor de saída **Y** e um vetor erro é calculado a partir do valor da saída desejada . Este erro é propagado para trás ("backpropagation") em direção à camada de entrada. A adaptação dos pesos se dá em função da contribuição das diversas sinapses para este erro (FREEMAN, 1991). O valor do passo de aprendizado μ, usualmente em torno de 0.25, determina a rapidez e a estabilidade da convergência. Um valor de μ muito pequeno torna muito lento o treinamento, enquanto um valor muito grande pode, mesmo, impedir a convergência. Uma boa estratégia é começar com um valor alto para μ e reduzi-lo à medida que o treinamento avança. O tempo necessário para o treinamento da rede depende fortemente dos pesos iniciais utilizados. Na seção 4, descreveremos um método para acelerar o treinamento, usando um algoritmo genético.

2.3 Redes Neuronais no Controle de Processos

As redes neuronais como os controladores nebulosos podem ser usados para representar o mapeamento entrada/saída de processos e desta forma constituem uma técnica bastante simples e eficaz de controle de processos não lineares complexos. O tempo de processamento é sempre um fator limitante em aplicações de tempo-real. Nos controladores nebulosos, este problema pode ser atenuado fazendo-se uso das tabelas de consulta ("look-up table"). Quando o número de antecedentes das regras e a partição nebulosa é muito fina, estas tabelas se tornam muito grandes, ocupando um excessivo espaço de memória. As redes neuronais, uma vez treinadas, fornecem de imediato a resposta para o sistema com uma exigência mínima de memória. Estas duas características, rapidez e pouca memória requerida, tornam as redes neuronais uma opção interessante para uma aplicação como os controladores nebulosos de tráfego (ARAÚJO, 1994). Nesta aplicação, procura-se determinar a extensão do tempo de verde da via com direito de passagem em função da taxa de chegada de veículos nesta via e da fila que se forma na outra via. A partir de dados gerados pelo controlador nebuloso de tráfego, uma rede neuronal backpropagation é treinada . As entradas são *Chegadas* e *Fila* de veículos. A saída é a *Extensão* de tempo do sinal verde. Depois de treinada, a rede pode substituir o módulo de inferência nebulosa. Os resultados que obtivemos foram muito bons. Os tempos de processamento foram bastante reduzidos e a fase de treinamento foi relativamente rápida para o conjunto de dados utilizados.

Numa outra aplicação recente (PATARO, 1993), uma rede neuronal backpropagation é utilizada para o cálculo da abertura entre os cilindros de um laminador. As entradas são a carga prevista, a largura da chapa e a espessura desejada. A saída da rede é a abertura entre os cilindros. A utilização da rede simplificou sobremaneira o cálculo da

abertura, tornando desnecessária a obtenção do módulo de rigidez.

3 Algoritmos Genéticos

São algoritmos de procura de soluções ótimas de finalidade geral. Eles utilizam mecanismos similares aos que regem a evolução das populações de seres vivos [GREFENSTETTE, 1993]. A rede neuronal backpropagation funciona bem desde que o espaço de solução não seja muito convoluído, pois neste caso existirão muitos mínimos locais, os quais retardam ou paralisam o processo de obtenção da solução ótima global. Os algoritmos genéticos não envolvem minimização de gradiente. Eles não são, pois, afetados por mínimos locais e constituem-se numa alternativa atraente para solução desta classe de problemas. Apresentaremos, abaixo, um método para acelerar a fase de treinamento das redes neuronais usando um algoritmo genético para inicializar os pesos da rede.

3.1 Descrição do método

Geramos aleatoriamente "**n**" vetores de pesos dentro de um intervalo onde supomos estar a solução para o problema. O conjunto de pares de teste entrada/saída é então aplicado à rede com cada um dos "**n**" vetores de pesos, e o erro quadrático é armazenado em cada caso. Os vetores de pesos são dispostos em ordem crescente do erro associado. Os 25% melhores vetores são duplicados, os 25% piores são eliminados e os 50% restantes, mantidos. O total dos vetores permanece o mesmo mas os 25% melhores dobraram sua incidência. Em seguida, algumas partes dos vetores são modificadas em um processo conhecido como "**crossover**". O número de vetores submetidos ao "crossover" é um parâmetro arbitrado, não existindo uma regra geral para definição de seu valor. Em nosso trabalho, trocamos dois pesos em posições aleatórias dentro de cada vetor [JANSON, 1993]. A fase final, chamada de "**mutação**", consiste em substituir, aleatoriamente, dentro da matriz de pesos alguns de seus valores. A taxa de mutação é também um parâmetro arbitrário. O processo de mutação injeta informação nova na população, o que é conveniente, pois não existe garantia de que a solução do problema esteja no universo dos pesos vigente. Por ser aleatória, a mutação pode também destruir um bom vetor de pesos antes que este possa ser duplicado. Na prática, observa-se que uma taxa elevada de mutação provoca uma oscilação nos valores dos erros a cada geração. O processamento genético dos dados a cada geração envolve as segintes fases:

- gerar aleatoriamente uma matriz de pesos dentro de um intervalo fixado;
- avaliar o desempenho de cada linha da matriz de pesos;
- duplicar as melhores e eliminar as piores.
- executar o "crossover";
- aplicar a mutação.

Repete-se o processo do passo 2 ao passo 5 até atingir o número fixado de gerações ou a precisão estipulada.

4 Um acelerador genético para a rede backpropagation

Testes realizados com o algoritmo genético mostraram que nos primeiros ciclos a adaptação é mais rápida do que no método backpropagation. O tempo gasto a cada ciclo no caso do genético é constante e corresponde aproximadamente ao produto de três fatores: o tempo gasto para uma

apresentação, o número de padrões de teste e o número de linhas da matriz de pesos. Após um avanço inicial rápido, o algoritmo genético se torna muito lento com melhorias ocorrendo após longos intervalos e devidas ao processo de mutação. No backpropagation, os tempos necessários para treinamento de cada par são variáveis, muito longos no início, decrescendo exponencialmente com a evolução do treinamento dos pesos. Estes fatos sugerem a aplicação de um método misto de treinamento no qual o método genético é utilizado nos primeiros ciclos, agilizando assim o treinamento da rede backpropagation. Em nossa aplicação, os pesos obtidos após 10 ciclos do algoritmo genético foram passados para o algoritmo backpropagation. Os resultados são mostrados, a seguir.

5 Resultados

O treinamento de uma rede backpropagation para o problema do controle de tráfego foi relativamente rápido. Julgamos mais interessante, para avaliar o desempenho do algoritmo genético, a aplicação da rede backpropagation para o cálculo da abertura entre os cilindros de um laminador. Os dados foram extraídos de [PATARO, 1993]. Fixamos a taxa de mutação em 0,5 e o passo de aprendizado em 0,25 e realizamos uma série de simulações para avaliar o ganho em se acoplar um algoritmo genético a uma rede backpropagation. Os resultados tabelados abaixo indicam o bom desempenho do algoritmo misto (genético e neuronal). Em todas as simulações realizadas houve uma redução significativa do número de iterações e do tempo de processamento. Outras aplicações vão ser utilizadas visando uma melhor compreensão da influência dos diversos parâmetros.

Série	Algoritmo misto	Backpropagation
1	7229 iterações	46484 iterações
2	9354 iterações	40481 iterações
3	21212 iterações	48530 iterações
4	23007 iterações	45450 iterações
5	2156 iterações	38649 iterações

Tabela 1

6 Conclusões

Os bons resultados obtidos com a utilização dos algoritmos genéticos para obtenção dos pesos iniciais de uma rede backpropagation, embora não definitivos, representam um passo importante para a viabilização das redes neuronais em aplicações tempo-real. As potencialidades dos algoritmos genéticos merecem ser mais cuidadosamente exploradas. O número de gerações e a taxa de mutação são apenas alguns dos parâmetros cuja influência deve ser investigada.

7 Referências Bibliográficas

FREEMAN, 1991 "Neural Networks". FREEMAN, J.A & SKAPURA, D.M. Addison Wesley, 1991.

GREFENSTETTE, 1993 "A Brief Tutorial". GREFENSTETTE, J.J. IEEE Expert, pp 6 - 9, 10/93.

HEBB, D.O., 1949 "The Organization of Behaviour". HEBB, D.O. John Wiley, N.Y., 1949.

HOLLAND, 1975 "Adaptation in Natural and Artificial Systems". HOLLAND, J. H. MIT Press, Cambridge, MA, 1975.

HOPFIELD, 1982 "Neural Networks and Physical Systems with emergent collective computational abilities". HOPFIELD, J.J. Proc. of the Nat. Acad. Sciences, Vol. 79, pp. 2554 - 2558, 1982.

JANSON, 1993 "Training Product Unit Neural Networks with Genetic Algorithms". JANSON, D. J. & FRENZEL, J. F. IEEE Expert, pp 26 - 33, 10/93.

KOSKO, 1992 "Neural Networks and Fuzzy Systems". KOSKO, B. Prentice Hall, 1992.

LEE, 1990 "Fuzzy Logic in Control Systems: Fuzzy Logic Controller". LEE, C.C. IEEE Transactions on Systems, Man and Cybernetics vol. 20, pp 419-434, 1990.

LIPPMAN, 1987 "An Introduction to Computing with Neural Nets". LIPPMANN, R.P. IEEE ASSP Magazine, pp. 4-22, 04/87.

McCULLOCH, 1943 "A Logical calculus of the ideas immanent in nervous activity". McCULLOCH, W.S. & PITTS. W. Bulletin of Math. Biophysics, vol. 4, pp 115-133, 1943.

PATARO, 1993 "Posicionamento Automático de Cilindros em Laminadores Empregando Treinamento com Redes Neuronais". PATARO, C. D. M. & SCHMIDT, W. G. & RESENDE, P. & HELMAN, H. III Seminário de Automação na Indústria, ABM, 1993.

1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajuba
Itajuba, 24 a 27 de outubro de 1994

NEURÔNIO COM ENTRADA QUADRÁTICA E CENTRO DE ATIVAÇÃO

A. L. C. Canella e L. P. Caloba

COPPE / EE / UFRJ
CP 68504 CEP 21945-970, Rio de Janeiro, Brasil
e.mail : caloba@coe.ufrj.br

RESUMO

Neste trabalho é apresentada uma configuração alternativa para a entrada do neurônio do tipo perceptron. Com esta entrada e utilizando-se um treinamento com erro-retro-propagado modificado, o neurônio é habilitado a reconhecer rapidamente classes cujas distribuições geométricas no espaço de entrada sejam linear ou esfericamente separáveis.

1. INTRODUÇÃO

Sabe-se que a rede neural de uma camada composta de neurônios do tipo perceptron só é capaz de separar classes cujas distribuições no espaço de entrada sejam linearmente separáveis[1]. Com a arquitetura multicamada, treinada pelo algoritimo do erro-retro-propagado, esta limitação não existe[1], mas a rede se torna mais complexa e o treinamento mais lento.

É apresentada neste trabalho uma configuração alternativa para a entrada do perceptron que habilita a rede neural de uma camada a reconhecer classes cujas distrubuições no espaço de entrada sejam linear ou esfericamente separáveis. Esta nova configuração é composta de duas alterações na entrada do modelo de perceptron: a primeira é a inclusão de uma nova entrada, denominada de entrada quadrática; a segunda é a inclusão da informação do centro de ativação, que é o baricentro das entradas englobadas pela superfície hiperesférica.

Para se obter o centro de ativação é necessário utilizar um outro algoritimo de treinamento em paralelo com o do erro-retro-propagado, sendo o algoritimo do erro-retro-propagado usado para alterar os valores das sinapes, enquanto que o outro algoritimo é utilizado para obter o baricentro das entradas a serem englobadas.

2. MODELO

Neste tópico é apresentado o modelo do neurônio com entrada quadrática e centro de ativação. Na figura 1 é mostrado o modelo e em seguida é definida a nomenclatura empregada.

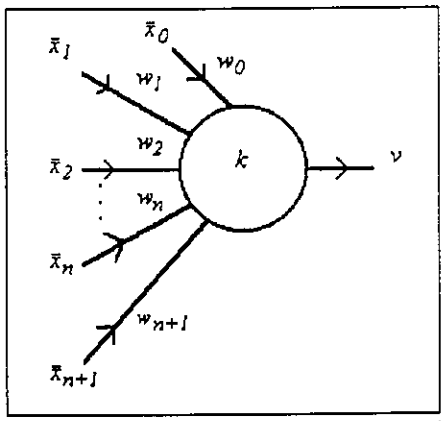


Figura 1 : Neurônio com entrada quadrática e centro de ativação.

Nomenclatura:

v : saída;
u : variável auxiliar;
k : ganho;
n : dimensão da entrada;
x_0 : entrada de polarização fixa em +1;
w_0 : sinapse de polarização;
$x_1, x_2, \ldots, x_n$: valores dos sinais de entrada;
$w_1, w_2, \ldots, w_n$: valores das sinapses;

x_{n+1} : entrada quadrádica;
w_{n+1} : sinapse da entrada quadrádica;
x : vetor de entrada;
w : vetor sinapse;
f(.) : função de ativação;

A equação (1) define as coordenadas do centro de ativação como o valor esperado das entradas da classe a ser separada. As equações (2) e (3) definem respectivamente as novas entradas e a entrada quadrádica. As equações (4) e (5) definem respectivamente a variável auxiliar u e a saída v, onde a função de ativação adotada é a função tangente hiperbólica.

$$(1) \qquad m_i = E(x_i) \quad \forall x_i \mid v = 0$$

$$(2) \qquad \bar{x}_i = x_i - m_i \,, \quad \text{p/} \quad i = 0,\cdots,n$$

$$(3) \qquad \bar{x}_{n+1} = \sum_{i=0}^{n} (x_i - m_i)^2$$

$$(4) \qquad u = \sum_{i=0}^{n+1} \bar{x}_i w_i$$

$$(5) \qquad v = tgh(u)$$

Aplicando (2) e (3) em (4) obtem-se a equação (6), que igualada a zero fornece a equação da "superfície separadora" da classe.

$$(6) \qquad u = \sum_{i=0}^{n} (x_i - m_i) w_i + w_{n+1} \sum_{i=0}^{n} (x_i - m_i)^2$$

Denomina-se de superfície separadora a superfície formada pelos pontos do espaço das entradas que anulam a variável auxiliar u dos neurônios da camada de saída.

3. NEURÔNIO COM ENTRADA LINEAR

O modelo do neurônio com entrada linear é um caso particular do modelo apresentado, onde a sinapse da entrada quadrática e as coordenadas do centro de ativação possuem valores nulos. Da equação (6) obtem-se a equação (7) que determina o valor da variável auxiliar neste modelo de neurônio.

$$(7) \qquad u = x_0 w_0 + x_1 w_1 + x_2 w_2 + \cdots + x_n w_n$$

Igualando a equação (7) a zero obtem-se a forma da superfície separadora. No espaço aumentado, espaço de dimensão $n+1$ obtido quando se considera a entrada de polarização x_0 livre, a superfície separadora é um hiperplano de ordem $n+1$ que passa necessariamente pela origem. No espaço de entrada considerado como um subespaço do espaço aumentado obtido quando se fixa a entrada de polarização x_0 em 1, a superfície separadora é um hiperplano do ordem n que não passa necessariamente pela origem.

É dado a seguir um exemplo hipotético em duas dimensões que pode ser projetado em uma dimensão sem perda de generalidade. Desta forma, é possível visualisar num plano o espaço de entrada ou o aumentado conforme for o caso.

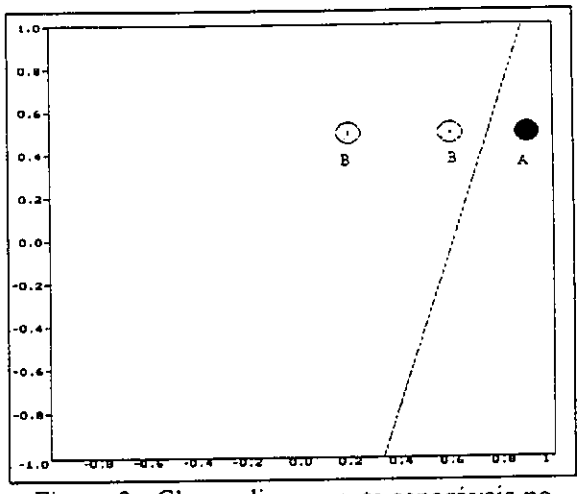


Figura 2 : Classes linearmente separáveis no espaço de entrada.

Na figura 2 é mostrado num plano o espaço de entrada com a disposição geométrica das classes a serem identificadas do sistema em duas dimensões. Na mesma figura é desenhada a reta que separa a classe A da classe B.

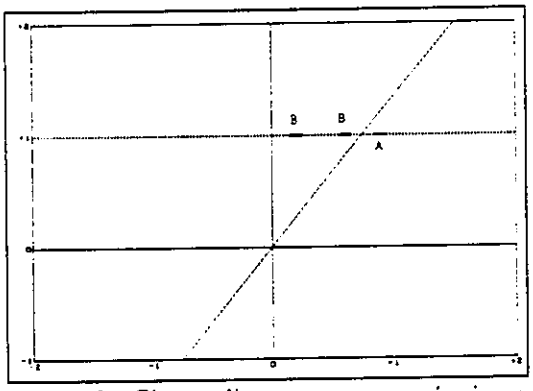


Figura 3 : Classes linearmente separáveis no espaço aumentado.

Na figura 3 é mostrado num plano o espaço aumentado do sistema em uma dimensão com a disposição geométrica das classes a serem identificadas. Na mesma figura é desenhada a

reta que passa necessariamente pela origem e cuja interseção com o subespaço das entradas determina um ponto que separa a classe A da B no espaço de entrada.

4. NEURÔNIO COM ENTRADA QUADRÁTICA

Neste tópico é apresentado um outro caso particular do modelo proposto, onde apenas as coordenadas do centro de ativação são nulas. Da equação (6) obtem-se a equação (8) que define a variável auxiliar u para este caso.

$$(8) \qquad u = \sum_{i=0}^{n+1} x_i w_i = \sum_{i=0}^{n} x_i w_i + w_{n+1} \sum_{i=0}^{n} x_i^2$$

Ao introduzir a entrada quadrática no modelo do neurônio linear modifica-se a forma da superfície separadora, que pode ser, conforme o valor da sinapase correspondente, um hiperplano ($w_{n-1}=0$) ou uma superfície hiperesférica ($w_{n-1} \neq 0$). Também, conforme o sinal da sinapse correspondente, é possível que o neurônio defina se será a parte interior ou exterior da superfície hiperesférica que fornecerá saída positiva. Deste modo, é possível que o neurônio identifique ou exclua uma determinada região esférica.

Igualando a equação (8) a zero obtem-se a forma da superfície separadora no espaço aumentado. Manipulando algebricamente a equação resultante obtem-se a equação (9), que é a forma de uma superfície hiperesférica. Da onde se obtem, após algumas manipulações algébricas, as equações (10) e (11) que definem respectivamente o raio r o as coordenadas c_i do centro da superfície hiperesférica separadora no espaço aumentado.

$$(9) \qquad u = -\frac{1}{w_{n+1}}\left[\sum_{j=0}^{n}\left(\frac{w_j}{2w_{n+1}}\right)^2 - \sum_{j=0}^{n}\left(x_j + \frac{w_j}{2w_{n+1}}\right)^2\right]$$

$$(10) \qquad r = \sqrt{\sum_{j=0}^{n}\left(\frac{w_j}{2w_{n+1}}\right)^2}$$

$$(11) \qquad c_i = -\frac{w_i}{2w_{n+1}}$$

Observa-se na equação (9) que o sinal da sinapse correspondente à entrada de polarização determina se será o interior ou o exterior da superfície hiperesférica que fornecerá saída positiva. Quando esta sinapse tem um valor positivo, o exterior fornece saída positiva. Caso contrário, é o interior que fornece saída positiva.

Fazendo x_0 igual a 1 e igualando a equação (8) a zero obtem-se a forma da superfície separadora no espaço de entrada. Manipulando algebricamente a equação resultante obtem-se a equação (12), que é a forma de uma superfície hiperesférica. Após algumas manipulações algébricas, obtem-se da equação (12) a definição matemática do raio r' e das coorcenadas c_i' do centro da superfície hiperesférica separadora no espaço de entrada, dadas respectivamente pelas equações (13) e (14).

(12)

$$u = -\frac{1}{w_{n+1}}\left[\sum_{i=1}^{n}\left(\frac{w_i}{2w_{n+1}}\right)^2 - 1 - \sum_{i=1}^{n}\left(x_i + \frac{w_i}{2w_{n+1}}\right)^2\right]$$

$$(13) \qquad r' = \sqrt{\sum_{i=1}^{n}\left(\frac{w_i}{2w_{n+1}}\right)^2 - 1}$$

$$(14) \qquad c_i' = -\frac{w_i}{2w_{n+1}}$$

Verifica-se da equação (13) a possibilidade da hiperesfera de ordem n possuir raio complexo no espaço de entrada. Isto apenas significa que a hiperesfera de ordem $n+1$ no espaço aumentado não intercepta o subespaço que define o espaço de entrada.

É dado a seguir o mesmo exemplo do tópico anterior com outra definição de classes. Na atual configuração o sistema não é mais linearmente separável, porém, o sistema é esfericamente separável.

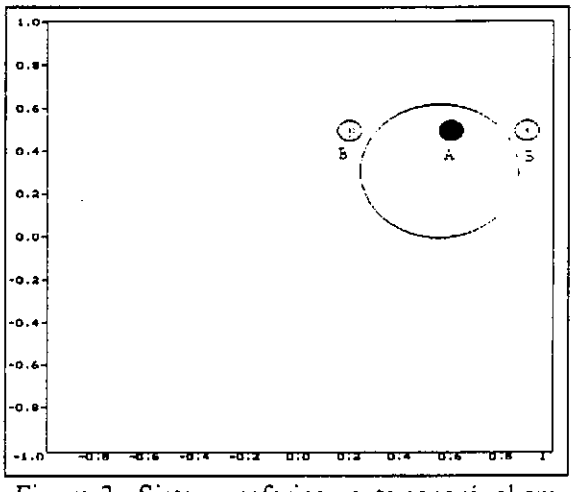


Figura 3 : Sistema esfericamente separável em duas dimensões.

Na figura 3 é mostrado num plano o espaço de entrada com a distribuição geométrica das classes a serem identificadas. Nota-se a impossibilidade de se realizar o classificador com uma rede neural de apenas uma camada, devido à distribuição geométrica das classes no espaço de entrada não ser separável por um hiperplano [1]. Entretanto, utilizando a entrada quadrática torna-se possível implementar este classificador com uma rede neural de apenas uma camada com apenas um neurônio. Na mesma figura é mostrada a superfície separadora traçada pelo neurônio, que é um círculo.

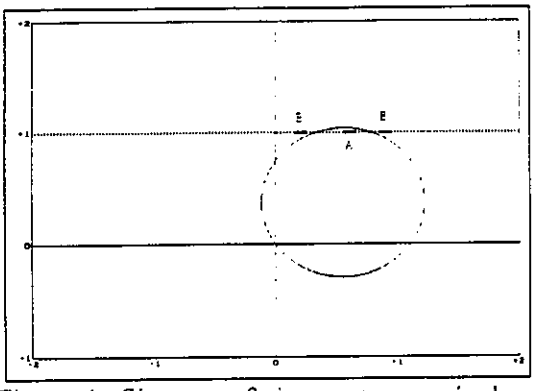


Figura 4 : Sistema esfericamente separável em uma dimensão.

Na figura 4 é possível visualizar o espaço aumentado e a distribuição geométrica das classes do sistema unidimensional. Nota-se que o círculo gerado pelo neurônio passa necessariamente pela origem no espaço aumentado. Este, intercepta o subespaço das entradas em dois pontos, que determinam a faixa na qual a classe A esta contida.

5. NEURÔNIO COM ENTRADA QUADRÁTICA E CENTRO DE ATIVAÇÃO

Com a introdução do centro de ativação, que no caso da rede ser de uma camada é o baricentro da classe a ser identificada, a superfície hiperesférica não tem que passar necessariamente pela origem no espaço aumentado. Fato este que acelera o treinamento por ter a hiperesfera um ponto fixo diretamente relacionado com a disposição geométrica das entradas a serem englobadas. Neste caso, a hiperesfera passa pelo ponto definido pelas coordenadas do baricentro da classe a ser identificada no espaço de entrada fazendo a coordenada referente a entrada de polarização nula.

Manipulando algebricamente a equação (6) obtem-se a equação (15) da qual pode-se extrair o raio r e as coordenadas c_i do centro da hiperesfera de ordem $n+1$, no espaço aumentado, representados pelas equações (16) e (17), respectivamente.

(15)

$$u = -\frac{1}{w_{n+1}}\left\{\sum_{i=0}^{n}\left(\frac{w_i}{2w_{n+1}}\right)^2 - \sum_{i=0}^{n}\left[x_i + \left(\frac{w_i}{2w_{n+1}} - m_i\right)\right]^2\right\}$$

(16) $$r = \sqrt{\sum_{i=0}^{n}\left(\frac{w_i}{2w_{n+1}}\right)^2}$$

(17) $$c_i = -\frac{w_i}{2w_{n+1}} + m_i$$

Por definição m_0 é uma constante nula e x_0 é uma constante unitária. Substituindo estes valores na equação (15) obtem-se a forma da superfície separadora no espaço de entrada, que é uma hiperesfera de ordem n. Esta é definida matematicamente pela equação (18), com raio r' e coordenadas c_i' definidos pelas equações (19) e (20), respectivamente.

(18)

$$u = -\frac{1}{w_{n+1}}\left\{\sum_{i=1}^{n}\left(\frac{w_i}{2w_{n+1}}\right)^2 - \frac{w_0}{w_{n+1}} - 1 + \sum_{i=1}^{n}\left[x_i + \left(\frac{w_i}{2w_{n+1}} - m_i\right)\right]^2\right\}$$

(19) $$r' = \sqrt{\sum_{i=1}^{n}\left(\frac{w_i}{2w_{n+1}}\right)^2 - \frac{w_0}{w_{n+1}} - 1}$$

(20) $$c_i' = -\frac{w_i}{w_{n+1}} + m_i$$

Para exemplificar é utilizado o mesmo exemplo do tópico anterior com a mesma definição de classes que torna à classe A não linearmente separável, porém esfericamente separável.

Na figura 5 é mostrado num plano a disposição geométrica das classes no espaço de entrada. Na mesma figura é mostrada a superfície separadora gerada pelo neurônio, que é um círculo.

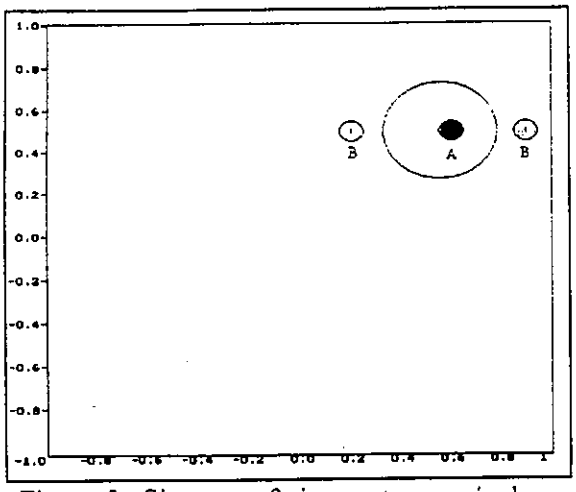


Figura 5 : Sistema esfericamente separável em duas dimensões com centro de ativação.

Na figura 6 é mostrado o espaço aumentado e a distribuição das classes no subespaço normal do sistema unidimensional. Pode-se visualisar a superfície separadora, que é um circulo, não passando pela origem e interceptando o subespaço das entradas em dois pontos. Este pontos definem a faixa onde está contida a classe A no espaço de entrada.

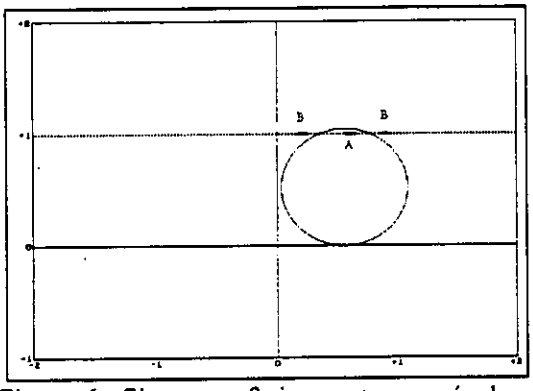


Figura 6 : Sistema esfericamente separável em uma dimensão com centro de ativação.

6. REDE NEURAL MULTICAMADA

Com a rede neural multicamada é possível realizar sistemas classificadores cujas distribuições das classes formadoras não sejam separáveis por hiperplanos[1]. Com a introdução da configuração alternativa para a entrada do neurônio do tipo perceptron nos neurônios da primeira camada intermediária. é possível obter grandes simplificações na estrutura da rede.

É dado a seguir um exemplo não possível de ser resolvido com uma rede neural de apenas uma camada, mesmo que se utilize a configuração alternativa para as entradas dos neurônios.

Observando a disposição geométrica das classes a serem identificadas pode-se claramente notar não se tratar de um sistema linearmente separável. Logo, a princípio seria necessário utilizar pelo menos uma camada intermediária para ser possível realizar este classificador com uma rede neural composta por neurônios do tipo Perceptron com entrada normal. Também, por haver necessidade de formar duas regiões fechadas, para poder separar a classe A da B, a princípio seria necessário utilizar pelo menos seis neurônios na camada intermediária. Portanto, observando a geometria das classes a serem identificadas, possível de ser feito por se tratar de um exemplo hipotético, conclui-se que a estrutura mínima necessária, porém, possa ser não suficiente, é uma rede com duas camadas com: seis neurônios na camada intermediária e um na de saída.

Um sistema classificador para o caso acima foi implementado por uma rede neural com uma camada intermediária com dois neurônios, utilizando-se a configuração alternativa para a entrada dos neurônios desta camada, e com apenas um neurônio, com entrada linear, na camada de saída. Neste exemplo, os neurônios não utilizam a informação do centro de ativação.

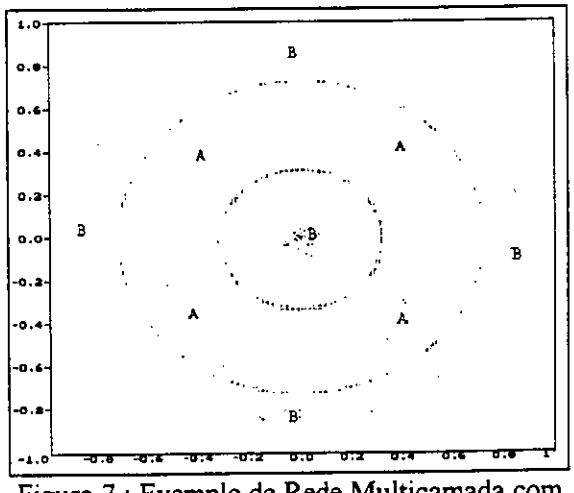


Figura 7 : Exemplo de Rede Multicamada com entrada quadrática.

Na figura 7 é mostrado o plano que define o espaço de entrada com a disposição geométrica das classes que definem o sistema classificador. Também é mostrada na figura a superfície separadora gerada pela rede neural, que é composta por dois circulos.

7. CONSIDERAÇÕES FINAIS

Pode-se verificar que o uso da entrada quadrática, que aumenta de uma unidade a dimensão da entrada, torna o neurônio do tipo perceptron mais versátil, com capacidade de formar, através de treinamento, superfícies separadoras no formato de calotas esféricas com qualquer nível de curvatura. Podendo, inclusive, formar uma superfície esférica fechada que identifica ou exclua uma região do espaço de entrada.

O uso da entrada quadrática só faz sentido quando os sinais de entrada não são normalizados, pois caso contrário a entrada quadrática seria uma constante. Devido à isto, numa estrutura multimacadas o uso da entrada quadrática só faz sentido nos neurônios da primeira camada intermediária, pois conforme a rede é treinada na maioria dos casos as saídas dos neurônios da camada intermediária tendem a saturação e normalização.

A introdução do centro de ativação, que acarreta no acréscimo de n parâmetros em cada neurônio, onde n é a dimensão da entrada, torna o neurônio ainda mais versátil, pois quebra a condição obrigatória de no espaço aumentado a superfície separadora ter que passar pela origem. Fato este indiferente quando se trata apenas de hiperplanos, porém essencial quando se utiliza superfícies hiperesféricas.

A determinação do centro de ativação, que é o baricentro das entradas englobadas pela superfície hiperesférica, só é direta em redes de uma camada. Neste caso, o centro de ativação é o baricentro da classe a ser identificada. Para redes multicamadas, como não se sabe a priori quais são as entradas que cada neurônio da primeira camada intermediária irá englobar, a determinação do centro de ativação torna-se bastante complexa. Uma solução possível é direcionar no início do treinamento todos os centros de ativação para o baricentro de todo o sistema. No decorrer do treinamento, conforme as saídas dos neurônios da primeira camada intermediária forem se saturando, encaminha-se o centro de ativação de cada neurônio para o baricento das entradas por eles englobadas. O algorítimo de treinamento das coordenadas do baricentro é dado pela equação (21), onde $\mathbf{m}$ é o vetor de coordenadas, β é a taxa de treinamento e k é o contador de iteração, idêntico ao treinamento de uma rede counterpropagation.

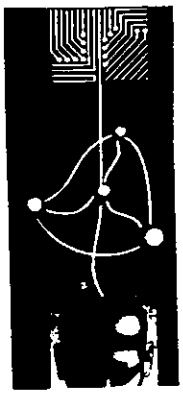
$$\mathbf{m}(k+1) = (1-\beta)\mathbf{m}(k) + \beta\mathbf{x}(k) \quad (21)$$

8. CONCLUSÕES

A introdução de uma entrada quadrática no neurônio tipo perceptron aumenta consideravelmente a sua capacidade de classificação com um pequeno aumento da sua complexidade, e uma redução na complexidade global da rede. O treinamento, entretanto, pode ficar mais lento, mas esta desvantagem pode ser superada usando-se o treinamento com centro de ativação.

9. REFERÊNCIA

[1] J. Hertz, A. Krogh e R. G. Palmer, "Introduction to the theory of neural computation", Santa Fe Institute Editorial Board, 1990.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá. 24 a 27 de outubro de 1994

Equalization of the Training Set for Backpropagation Networks Applied to Classification Problems

Frederico dos Santos Liporace, Ricardo José Machado and Valmir C. Barbosa

Abstract— **One of the problems faced by multilayer perceptrons trained by the backpropagation algorithm when applied to classification problems is the low sensitivity of the resulting network to classes statistically less represented in the training set. This paper proposes that in such cases it is better to build a modular network, assigning to each independent module the task of recognizing one specific class and rejecting the others. The use of a modular architecture enables the construction of a modified training set for each module that tries to minimize the problem of the less represented classes. This modification or *equalization* assigns a *relevance degree* to each training sample. This gives each sample a different degree of importance, and has the same effect of the replication of some samples in the training set. This technique was applied in an application related to satellite imagery classification, and the obtained results show that the modules trained with the equalized training set exhibit far better results than others trained with the "plain" training set.**

Keywords— **Neural networks, backpropagation, classification.**

I. Introduction

The error backpropagation algorithm is by far the most used to build neural-network based applications nowadays. Contrasting to the ease that it is applied to construct neural networks able to solve "toy problems", there is a great amount of effort involved when designing a larger network required for some practical applications.

This paper describes one problem frequently found during the design of classifiers using multilayer perceptrons trained with the backpropagation algorithm, namely the low sensitivity of the classifier to classes less represented (prevalent) in the training set.

Frederico dos Santos Liporace and Ricardo José Machado are with IBM Rio Scientific Center, Caixa postal 4624, 20001-970, Rio de Janeiro - RJ, Brazil. e-mail:liporace@vnet.ibm.com

Valmir C. Barbosa is with Universidade Federal do Rio de Janeiro, Caixa postal 68511, 21945-970, Rio de Janeiro - RJ, Brazil.

The proposed solution is to build a modular network architecture as well as a different equalizations of the same training set. The resulting equalized training sets are then used to train each module separately.

This modular architecture consists of network modules with independent hidden and output layers, each one assigned to the task of recognizing one particular class and rejecting the others. Besides other advantages discussed later, this modular architecture permits the training set to be suitable equalized for each module, as these are trained separately. This equalization intends to promote the equilibrium between the number of examples favoring the module's class and the number of examples against the module's class. This is achieved by the virtual replication of some examples in the training set.

The training set equalization technique proposed in this paper is discussed in the context of an application related to the interpretation of satellite imagery, but we believe that it may be useful in the design of many others neural network based classifiers. Section II gives a short description of the application and the adopted neural network solution. Section III describes some important characteristics of the set used to train the neural network. Section IV describes the adaptive learning rate procedure employed to train the networks. Section V discusses the advantages of using a modular architecture, and why that was the selected option for use in our system. This Section describes also the procedure used to construct the equalized training set for each module. In Section VI the improvements obtained by the proposed modifications in the training set are presented, and the concluding remarks are made in Section VII.

II. The Classifier System

Our initial problem was to develop a neural-network based classifier to be applied in the automatic interpretation of Landsat-5 satellite images from the Amazon region. The main interest was to identify deforested areas such as road building, mining, agriculture and other kinds of human activity that could affect the environment. A detailed description of the problem, the adopted solution and the results can be found in [1], [2]. Here we present

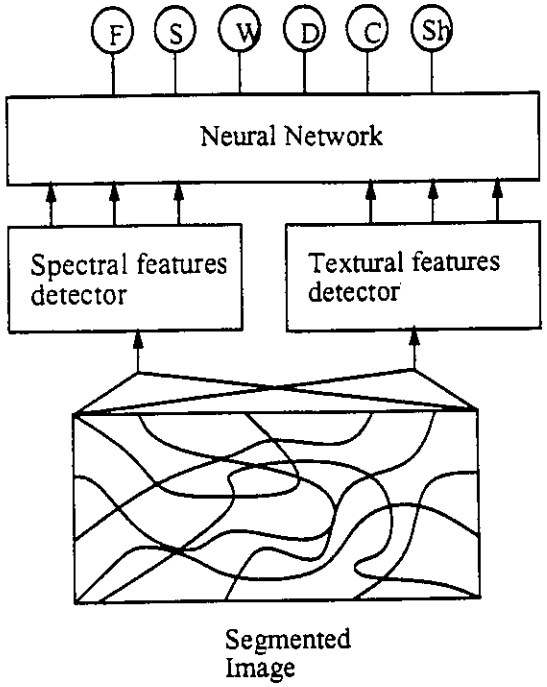


Fig. 1. Schematic of the proposed architecture

a short description of the system necessary to introduce the motivation of the equalization of the training set.

The classifier architecture is shown in Figure 1. In a first pass, the image generated by the satellite is partitioned in segments with homogeneous spectral characteristics. Later, a set of numeric attributes for each segment is calculated. These attributes consists of information related to spectral and textural features of a segment. We may cite as spectral features the gray level average of a segment in each spectral band monitored by the satellite, and as textural features the variance, entropy and correlation of each segment, as defined in [3]. These numerical attributes or *descriptors* of each segment are then presented to a feed-forward neural network trained by the backpropagation algorithm, which outputs the membership of the segment that is being analyzed in each of the defined categories. We employed a fuzzy classification approach, as it is allowed for a segment to have a partial degree of membership to more than one category. The conventional crisp classification is a particular case of the fuzzy one, where the segments are constrained to have a full membership to only one category.

There are four defined categories of interest in our classification problem, namely Forest (F), Savanna (S), Water (W) and Deforestation (D). These categories are called *basic categories*, as they embed all the relevant information to be monitored. There are also two *interfering categories*, namely Shadow (Sh) and Cloud (C), defined to account with the presence of interference caused by clouds and shadows in the images.

Situation	F	S	W	D	C	Sh
Deforested area	0	0	0	1	0	0
Deforested area with incipient reforestation	.25	0	0	.75	0	0
Cloud (opaque)	0	0	0	0	1	0
Tenuous shadow over forest	1	0	0	0	0	.5

TABLE I
EXAMPLES OF VALID CLASSIFICATIONS FOR A SEGMENT

The fuzzy classification approach allows to represent phenomena like the transition between two basic classes, such as the growth of forest in a area that was previously deforested but abandoned later, and the presence of interference like shadows and clouds that still permit the identification of the basic category of the segment, such as the presence of a transparent and thin cloud over a region of forest. Table I shows some examples of fuzzy classifications to clarify the idea.

The membership values of the segments in each category may vary in the interval $[0, 1]$, with 0 indicating empty and 1 indicating full membership of the segment in each category.

III. Characteristics of the Training Set

The database used to train and test the performance of the neural network is composed of five representative images of the Amazon region. These images were segmented by a region-growing technique [3], and the resulting segments were classified by an experienced photo-interpreter using the fuzzy-model approach described in Section II. To simplify the photo-interpreter's task, the allowed membership values were restricted to the set $\{0, 0.25, 0.5, 0.75, 1\}$.

This database is composed of approximately 17 thousand segments, from which two thirds were used to construct the training set of the network and the remaining third was used to measure the generalization ability of the trained network.

Table II shows the distribution of the segments that form the training set in respect to its classifications. Only the basic categories were considered, and a segment was considered to belong to the category that had the greatest membership among all the basic categories. Segments that were classified as having membership only in the cloud or shadow categories were not considered in this table, that's why the percentages doesn't sum up to 100%. As Table II shows, the distribution is highly unbalanced in favor of the F and D categories, leaving far less examples of the W and S ones.

F	S	W	D
38.76 %	2.03 %	3.65 %	40.34 %

TABLE II
DISTRIBUTION OF THE SEGMENTS IN RESPECT TO ITS CLASSIFICATIONS

IV. The Adaptive Backpropagation

The backpropagation algorithm defines an error function between the desired and the actual output of the network and then searches for a set w of synaptic weight values that minimizes this function by a steepest-descent procedure. The error function usually employed is

$$E(\mathrm{w}) = \frac{1}{2}\sum_{\mu=1}^{m}\sum_{i=1}^{n}[D_i^\mu - S_i^\mu]^2 \qquad (1)$$

for a training set of m samples and a network with n neurons in its output layer. We called the actual output of the i^{th} neuron for the μ^{th} sample S_i^μ, and the desired output for the same sample D_i^μ.

The derivatives of Equation 1 in respect to each weight of the network are then calculated and each weight w_{ij} is updated following the equations

$$\Delta w_{ij} = -\alpha\frac{\partial E}{\partial w_{ij}} \qquad (2)$$

$$w_{ij}^{\text{new}} = w_{ij}^{\text{old}} + \Delta w_{ij} \qquad (3)$$

where α is a positive constant known as *learning rate*, selected experimentally.

The backpropagation algorithm calculates the Δw_{ij} terms for each synapse as each sample in the training set is presented to the neural network. Shortly, we have

$$\Delta w_{ij}^p = \alpha\frac{\partial E^p}{\partial w_{ij}} \qquad (4)$$

where p is a sample from the training set and E^p is the error for the p sample. The synapses can be updated after the presentation of each sample (*stochastic* learning), applying the result of Equation 4 in Equation 3. This procedure updates the synaptic weight values to minimize the error for the specific pattern that is being presented to the network at the moment.

Another way to update the synaptic weight values is to accumulate all the Δw_{ij}^p terms for all samples in the training set and use this result in Equation 3, in what is called *batch* learning. This procedure tries to minimize the error for *all* the samples in the training set.

The large size of our training set makes it difficult to find good values for the learning rate parameter by trial and error, due to the long time required to run each trial. It was also verified in our early experiments that values that were good in the beginning of the training process caused instability later [2], which makes this search even harder and requires a continuous monitoring or the error rate evolution. We chose then to use an adaptive learning rate procedure, as suggested in [4].

This procedure can be summarized as follows: the weights are updated after the presentation of all the samples in the training set (batch learning), and the previous synaptic weight values are stored. This way it is possible to "undo" the update. After each update, the behavior of the error evolution is checked. The learning rate parameter is then modified following the rule:

$$\alpha^{\text{new}} = \alpha^{\text{old}} + \Delta\alpha$$

with

$$\Delta\alpha = \begin{cases} +a & \text{if the error decreases consistently} \\ -b\alpha & \text{if the error increases} \\ 0 & \text{otherwise} \end{cases}$$

where a and b are appropriate positive constants.

If the new error value exceeds the previous one, the process overshot and the learning rate is reduced. In our implementation, the synaptic weight values were also restored to the previous ones conveniently saved and the iteration was retried with the new learning rate until it succeeds in reducing the error. In the other way, if the error decreases consistently, the learning rate is increased in hope to achieve a faster convergence. We considered consistent a decrease of the error value for 10 consecutive iterations of the algorithm.

V. The Proposed Solution

The proposed solution to the low sensitivity of the network to classes less represented in the training set is composed of two parts: the first is the construction of a modular network, which permits then the training set for each module to be equalized in a way that the effect of the uneven distribution is reduced.

A. Why a Modular Network?

The first approach that one might consider when designing a neural network for the application described in this paper is to build a structure like that shown in Figure 2. Within this structure, the hidden layer is shared between all the neurons of the output layer, and each neuron on the output layer is assigned to one category.

In the first tests we made with that structure we found that the resulting network had a very low sensitivity to examples belonging to the S and W categories. One possible explanation for that behavior is that the backpropagation algorithm tries to minimize the *global* error function shown in Equation 1.

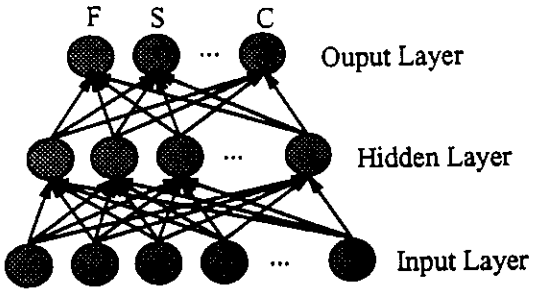


Fig. 2. A non-modular network

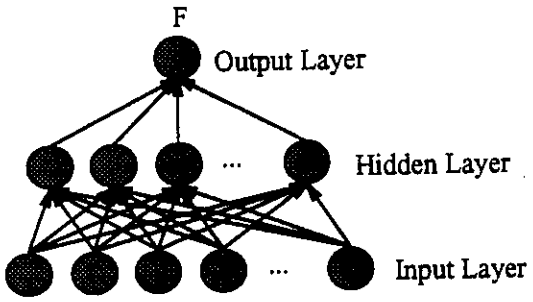


Fig. 3. A module associated to the F category

As the segments from the classes S and W have far less examples than that of the F and D categories, these two first are simply not considered as they have a very low influence on the global error.

Another way to build the network is to use a modular architecture. Figure 3 shows the structure of one of the modules. Each module is associated with a category, and the modules share only the input layer, maintaining independent hidden layers and an output layer with a single neuron. Each module is specialized in detecting one specific category and rejecting the others.

This second architecture has some very distinctive advantages over the first one. As there are no connections between the hidden layers, the modules can be trained separately, possibly in parallel using different machines. Moreover, one can select the modules that have a poor performance and concentrate the training on these modules, without affecting the others. This way, a more efficient use of the available computational power is allowed, specially because we observed that some modules trained significantly faster than others.

B. The Equalized Training Set

Another important advantage of the modular architecture is the possibility to adjust the training set of each module in a way that the effect of the unfair distribution between the classes is minimized. Each module of the network is specialized in detecting one category and rejecting the others. Hence, if we are concerned to one specific network module, we can classify the examples in the training set in two categories: the ones that are *in favor* of the module's class and the ones that are *against* the class, depending of the degree of membership in the module's class assigned to that examples by the photo-interpreter. One possible way to classify one example on the *in favor* or *against* categories is to check if the membership value for that example exceeds or not 0.5.

We can promote the equilibrium between these two antagonist categories by associating to each training set sample p a *relevance degree* $r(p)$. This relevance degree is then used to multiply the α term in Equation 4, yielding to the modified version:

$$\Delta w_{ij}^{p} = \alpha r(p) \frac{\partial E^{p}}{\partial w_{ij}} \tag{5}$$

We may construct for each module c a set of relevance values r^c that achieve the equilibrium between the *in favor* and *against* samples by setting initially unitary relevance values to all samples and then raise the relevance of samples that belongs to the less represented group, being that the *in favor* or *against* one. We implemented that raising by simply increasing the relevance values for all the samples of the less represented group by one and repeating this pass until the equilibrium is achieved.

Note that when batch training is used the proposed modification is equivalent to $r(p)$ presentations of the p sample during the calculation of the Δw_{ij} terms, that is, it has the same effect of the simple replication of the p sample $r(p) - 1$ times in the training set.

VI. Experimental Results

This section compares the performance of the modules trained with the "plain" training set and with an equalized version of the same training set. We chose to show the results for the W module, since a similar behavior was encountered for all the remaining modules. We implemented the adaptive backpropagation method described in Section IV in an IBM Risc 6000 - Model 560 workstation, and each training run presented required approximately two days to complete.

Figure 4 shows the sensitivity and specificity of the module as the training process is executed. This module was trained without an equalization of the training set. The sensitivity and specificity are measured every 600 epochs, and the dots in the figures represent the values at each measurement. It may be observed from Figure 4 that the values of the sensitivity are very low for this network, starting at 0.04 in the beginning of the training and raising to 0.35, still a low value, after 8400 epochs. There is a clear tendency of the network trained with the "plain" training set to wrongly reject examples that belong to the water category.

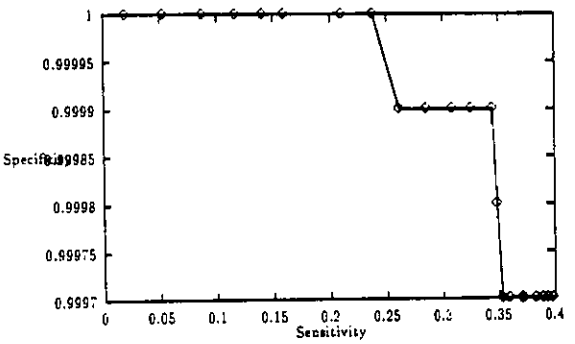


Fig. 4. Evolution of sensitivity and specificity to the W class for the "plain" training set

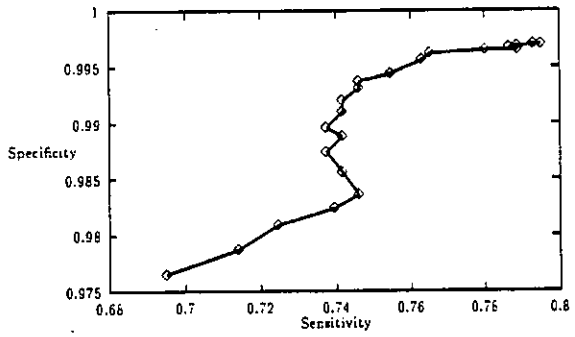


Fig. 5. Evolution of sensitivity and specificity to the W class for the equalized training set

The same module was trained again, now with an equalization of the training set as proposed in Section V-A. The use of the equalized training set clearly yielded to much larger values of sensitivity, as Figure 5 shows. After the first 600 epochs, the sensitivity value was already near 0.7, which is much better than the obtained after 8400 training epochs without the equalized training set. We may also note that the specificity value is still as high as it was before. As the training proceeds, both values still increase.

Another interesting behavior was observed during the training of the D module. Most of the errors made by this module were in S examples, since the spectral characteristics of this category are very similar to that of the D category. We also found that in that cases it is better to assign greater relevance values to S samples, as this makes the error function to be minimized by the backpropagation algorithm more sensitive to errors between S and D categories.

VII. Concluding remarks

One of the problems that the backpropagation algorithm exhibits when applied to classification problems is its low sensitivity to classes less represented in the training set. Our proposed solution to that problem consists in building a modular architecture, in which modules with different hidden layers are assigned the task to detect one class and reject the others. The use of this modular architecture enables the construction of an equalized training set for each module.

The proposed equalization has the same effect of simply replicating examples that belong to one less represented class in the training set, but without a significant increase in the training time.

The technique here described may be applied to fuzzy classification problems in general, and of course apply to the particular case of crisp classification. The learning should be preferably done in batch, as its allows the use of an adaptive learning rate procedure. This adaptive control of the learning rate and the possibility of training the network's modules in parallel on different machines are valuable advantages in complex problems domains that require long training times.

The results we obtained demonstrate that with the proposed modular architecture and the equalization of the training set for each module it is possible to construct networks that are more sensitive to classes less represented in the training set.

References

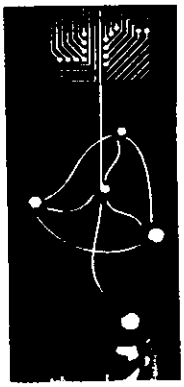
[1] Valmir Carneiro Barbosa, Ricardo José Machado, and Frederico dos Santos Liporace. A neural system for deforestation monitoring on Landsat images of the Amazon region. *International Journal of Abstract Reasoning*. To appear.

[2] Frederico dos Santos Liporace. Um sistema neural para monitoração do desflorestamento na região Amazônica utilizando imagens do Landsat. Master's thesis, Universidade Federal do Rio de Janeiro, 1994. In Portuguese.

[3] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice-Hall, 1982.

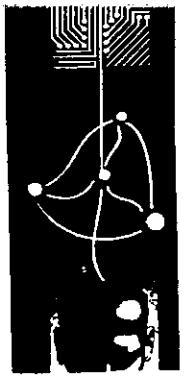
[4] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.

Anotações



AMBIENTES DE DESENVOLVIMENTO

Anotações



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajuba, 24 a 27 de outubro de 1994

Modelagem do Sistema de Inferência Difusa de Tsukamoto na Arquitetura Anfis[1]

André Gomes de Melo Medeiros
(agmm@di.ufpe.br)
Edson Costa de Barros Carvalho Filho
(ecdbcf@di.ufpe.br)
Departamento de Informática
Cx 7851, 50.732-970, Recife, PE, Brasil
Tel.: (081) 271-8430, Fax: (081) 271-4925

Sumário

Este artigo tem por objetivo modelar o sistema de inferência difusa definido por Tsukamoto através da arquitetura Anfis, de forma a incorporar a capacidade de aprendizado e de refinamento das regras difusas. Esta modelagem propicia, portanto, que regras difusas sejam obtidas a partir do conhecimento disponível de um especialista, bem como através de um conjunto de pares entrada-saída de treinamento conforme aplicado no processo de aprendizado das redes neurais. Por fim, uma simulação é realizada para demonstrar o grau de aproximação de uma função não-linear obtido pela rede Anfis que implementa a inferência difusa de Tsukamoto.

1 Introdução

Técnicas de controle difuso são aplicadas a sistemas onde a elaboração de um modelo matemático através de métodos clássicos da Teoria de Controle apresenta grande complexidade e sérias limitações. Grande parte das limitações decorre da necessidade de resolução de um conjunto de equações complexas, como por exemplo equações diferenciais de segunda ordem, e do fato dos sistemas sob consideração serem comumente mal-definidos, incertos e altamente não-lineares.

O controle difuso[6] consiste na construção de um modelo de acordo com o comportamento de um especialista que é capaz de controlar o sistema. As ações do especialista são especificadas na forma de regras linguísticas, que serão posteriormente transcritas em termos da teoria da lógica difusa provendo um cálculo para simular o comportamento do especialista.

Um ponto crítico é que a especificação de regras linguísticas adequadas depende do conhecimento e experiencia do especialista. Entretanto, a maior dificuldade é a não formalização do processo de transcrição das regras linguísticas em termos da teoria da lógica difusa. Isto implica na escolha arbitrária e subjetiva das funções de pertinência, o que pode influenciar sensivelmente a performance do controlador difuso. Desta forma, métodos de ajuste das funções de pertinencia são necessários.

Redes neurais se candidatam a resolver o problema do ajuste das funções de pertinencia, o que justifica o crescente interesse na combinação de redes neurais e sistemas difusos[3][4][5]. Do ponto de vista de redes neurais, esta combinação também traz vantagens, como será mostrado através da arquitetura Anfis[4], pois possibilita a aplicação de algum conhecimento disponível a priori sobre a estrutura da rede como forma de simplificar e acelerar o processo de aprendizagem. As redes neurais clássicas, como as derivadas do modelo de McCulloch-Pitts[7][9], não permitem a aplicação de conhecimento disponível a priori e, após treinadas, são normalmente entendidas como uma caixa-preta: não é possível, em geral, extrair qualquer tipo de conhecimento estrutural da rede neural treinada. Redes neurais e sistemas difusos se constituem em técnicas alternativas para aplicações como reconhecimento de padrões, interpolação de funções e controle.

Na seção 2, são discutidos alguns tipos de sistemas de inferencia difusa, dando especial enfase ao sistema de inferencia difusa de Tsukamoto. Na seção 3, é derivada uma rede Anfis que é equivalente ao sistema de inferencia difusa de Tsukamoto. Na seção 4, uma simulação é realizada para verificar o grau de aproximação de uma função não-linear pela rede Anfis equivalente ao sistema de inferencia difusa de Tsukamoto. Na última seção, conclusões são estabelecidas.

[1]Este trabalho contou com o apoio financeiro do CNPq e da FACEPE

2 Sistemas de Inferência Difusa

Sistemas de inferencia difusa, conhecidos também como controladores difusos quando aplicados em controle, são constituídos por um conjunto de regras IF-THEN difusas, funções de pertinencia associadas a termos linguísticos nas regras, e um mecanismo de inferencia denominado raciocínio difuso.

Existem vários tipos de sistemas de inferencia difusa proposto na literatura[1][2], como os formulados por Tsukamoto[11], Mandani[6], e Takagi e Sugeno[10]. Cada um destes tipos de sistema de inferencia difusa compreende diferentes tipos de regras IF-THEN difusas e raciocínio difuso. O sistema de inferencia difusa definido por Takagi e Sugeno, por exemplo, apresenta regras difusas da seguinte forma:

IF *deslocamento* é **pequeno** THEN *força* é 0.3

onde *deslocamento* é uma variável de entrada e *força*, uma variável de saída. O qualificador **pequeno** é um termo linguístico que denota um conjunto difuso. Neste tipo de inferencia difusa, termos linguísticos (conjuntos difusos) só podem ocorrer na premissa (parte IF) da regra.

O sistema de inferencia difusa proposto por Tsukamoto e o especificado por Mandani apresentam regras difusas da seguinte forma:

IF *deslocamento* é **pequeno** THEN *força* é **fraca**

onde *deslocamento* e *força* são variáveis de entrada e de saída, respectivamente. Os qualificadores **pequeno** e **fraca** são termos linguísticos que representam conjuntos difusos. Portanto, para os sistemas de inferencia difusa de Tsukamoto e Mandani, termos linguísticos (conjuntos difusos) ocorrem tanto na premissa (parte IF) como no consequente (parte THEN) da regra. Para os sistemas de inferencia difusa de Tsukamoto e Mandani, o consequente da regra atribui um conjunto difuso (**fraca**) à variável de saída (*força*), ao contrário do sistema de inferencia difusa de Takagi e Sugeno que atribui um valor numérico (0.3). Desta forma, para a obtenção do resultado final, os sistemas de inferencia de Tsukamoto e Mandani exigem a realização de um processo de *defuzzyficação* que visa converter o valor difuso (conjunto difuso) atribuído à variável de saída da regra para um valor numérico simples.

O mecanismo de raciocínio difuso de Mandani realiza o processo de *defuzzyficação* através do cálculo do centro de gravidade (centróide) da função de pertinencia do conjunto difuso atribuído à variável de saída. No entanto, este método apresenta certa vulnerabilidade porque depende da realização do cálculo de integrais, que é notoriamente não trivial.

Afim de evitar tais dificuldades, o raciocínio difuso de Tsukamoto exige que as funções de pertinencia dos termos linguísticos dos consequentes sejam monotonicas, de forma que um método de *defuzzyficação* não-convencional baseado nas funções inversas destas funções de pertinencia possa ser aplicado na avaliação das regras.

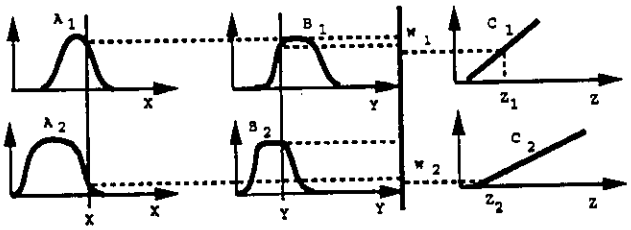
Para ilustrar o procedimento de raciocínio difuso de Tsukamoto, é assumido uma base de regras contendo exatamente duas regras IF-THEN difusas com duas variáveis de entrada x e y e uma variável de saída z. Considerando as regras do sistema de inferencia difusa:

Regra 1: IF x é A_1 and y é B_1 THEN z é C_1
Regra 2: IF x é A_2 and y é B_2 THEN z é C_2

onde A_1, A_2, B_1, B_2, C_1 e C_2 são termos linguísticos(**pequeno, médio, fraca, etc**). Por simplicidade, as funções de pertinencia dos termos linguísticos C_1 e C_2 dos consequentes das regras são restritas a funções lineares, que são monotonicas. Uma função de pertinencia linear, μ, é definida por

$$\mu(x) = \begin{cases} \frac{-x+a}{a-b}, & se\ (x \in [a,b] \wedge a < b) \\ & \vee\ (x \in [b,a] \wedge a > b) \\ 0, & se\ (x \notin [a,b] \wedge a < b) \\ & \vee\ (x \notin [b,a] \wedge a > b) \end{cases} \quad (1)$$

com $\mu(a) = 0$ e $\mu(b) = 1$.



$$Z = \frac{w_1 z_1 + w_2 z_2}{w_1 + w_2} = \overline{w}_1 z_1 + \overline{w}_2 z_2$$

Figura 1: Sistema de inferencia difusa de Tsukamoto com 2 regras

O mecanismo de raciocínio difuso de Tsukamoto é mostrado na Figura 1. Inicialmente, os graus de compatibilidade (graus de pertinencia) do valor de x em relação aos termos linguísticos A_1, A_2 e do valor de y em relação aos termos linguísticos

B_1 e B_2 são calculados. Esta etapa do processo de raciocínio difuso é conhecida como *fuzzyficação*.

A seguir, o grau de verdade da premissa da i-ésima regra é obtido através da operação AND da lógica difusa, usualmente efetuada multiplicando-se ou tomando-se o mínimo dos graus de pertinência dos valores contidos nas variáveis de entrada em relação aos conjuntos difusos definidos pelos termos linguísticos na premissa da regra:

$$\begin{aligned} w_i &= \mu_{A_i}(x_1)\mu_{B_i}(x_2), ou \\ &= min(\mu_{A_i}(x_1), \mu_{B_i}(x_2)) \end{aligned} \quad (2)$$

O valor z_i, resultante da avaliação da i-ésima regra, é obtido através da *defuzzyficação* baseada na função inversa da função de pertinência do termo linguístico do consequente da regra, sendo calculado pela equação:

$$z_i = \mu_{C_i}^{-1}(w_i) \quad (3)$$

A equação especifica que o resultado da avaliação de cada regra, z_i, deve ser um valor que possui grau de pertinência em relação ao termo linguístico C_i do consequente da regra igual a w_i, que é o grau de verdade da premissa da regra. Sendo μ_{C_i}, a função de pertinencia do termo linguístico C_i definida pela equação (1), a sua função inversa é dada por:

$$\mu_{C_i}^{-1}(x) = -x(a-b) + a = px + q \quad (4)$$

com $x \in [0,1]$, $p = b - a$ e $q = a$.

A saída geral do sistema de inferência difusa de Tsukamoto com n regras é calculada através da média ponderada da saída de cada regra, onde os pesos são os graus de verdade da premissa de cada regra:

$$z = \frac{\sum_{i=1}^{n} w_i z_i}{\sum_{i=1}^{n} w_i} = \frac{\sum_{i=1}^{n} w_i \mu_{C_i}^{-1}(w_i)}{\sum_{i=1}^{n} w_i} \quad (5)$$

3 Arquitetura Anfis

A arquitetura **Anfis**[4], um acrônimo para *Adaptative Network-based Fuzzy Inference System*, engloba uma classe de redes adaptativas que são funcionalmente equivalentes a tipos de sistemas de inferência difusa. Uma rede adaptativa é um tipo especial de rede *feedforward* multi-nível em que cada nó computa uma função específica. As conexões que interligam os nós numa rede adaptativa somente indicam a direção do fluxo de sinais que se propagam entre os nós, não havendo pesos associados a elas. A rede adaptativa é constituída por dois tipos de nós: os adaptativos, que possuem parametros que podem ser alterados pelo algoritmo de aprendizado, e os não-adaptativos que não possuem nenhum parametro. A capacidade de aprendizado da rede adaptativa decorre da presença dos parametros dos nós adaptativos.

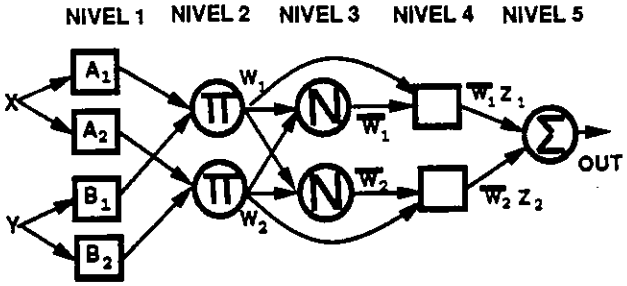


Figura 2: Anfis equivalente ao Sistema de Inferencia Difusa de Tsukamoto da Figura 1

A seguir, é derivada a estrutura dos cinco níveis de uma rede **Anfis**(Figura 1), que é equivalente a um sistema de inferência difusa de Tsukamoto. Denotando-se por O_i^k o i-ésimo nó do nível k, tem-se:

- **Nível 1**: Todo nó i neste nível é um nó adaptativo computando a função

$$O_i^1 = \mu_{A_i}(x), \quad (6)$$

onde x é a entrada para o nó i, e A_i é um termo linguístico (como **pequeno**, **grande**, **alto**, etc) que denota um conjunto difuso, sendo μ_{A_i} a função de pertinencia associada. Portanto, O_i^1 é a função de pertinência de A_i e especifica o grau que x satisfaz o conceito expresso por A_i. Usalmente a função μ_{A_i} é escolhida tendo a forma semelhante a uma distribuição guassiana com máximo igual a 1 e mínimo igual 0, como

$$\mu_{A_i} = \frac{1}{1 + \left[(\frac{x-\gamma_i}{a_i})^2\right]^{\beta_i}} \quad (7)$$

ou

$$\mu_{A_i} = exp\left\{-\left[(\frac{x-\gamma_i}{\alpha_i})^2\right]^{\beta_i}\right\} \quad (8)$$

onde$\{\alpha_i, \beta_i, \gamma_i\}$ é o conjunto de parâmetros do nó i. Alterando-se os valores destes parâmetros, provacam-se variações nas funções de pertinência do termo linguístico A_i. Funções diferenciáveis em intervalos e contínuas, tais como as que apresentam forma trapezoidal e triangular, são também qualificadas como candidatas para funções dos nós neste nível. Esses nós são responsáveis pelo processo de *fuzzyficação*. Parametros neste nível são referidos como **parâmetros das premissas**, em analogia ao fato de que em um sistema de inferencia difusa o resultado da *fuzzyficação* é usado para a avaliação do grau de verdade das premissas das regras.

- **Nível 2**: Todo nó neste nível é um nó não-adaptativo, que computa o produtório dos sinais recebidos de alguns dos nós do nível anterior:

$$O_i^2 = w_i = x_1 \times x_2 \times \ldots \times x_j. \quad (9)$$

onde x_k denota o k-ésimo elemento do vetor de entrada do nó, que recebe os sinais do nível anterior. A saída de cada nó neste nível corresponde ao grau de verdade da premissa de uma regra no sistema de inferencia difusa. A função dos nós deste nível é realizar a operação AND da lógica difusa, que neste caso é implementada através do produto.

- **Nível 3:** Todo nó neste nível é um nó não-adaptativo. O i-ésimo nó calcula a proporção do grau de verdade da i-ésima regra(saída do i-ésimo nó do nível anterior) e a soma dos graus de verdade de todas as n regras:

$$O_i^3 = \overline{w_i} = \frac{w_i}{\sum_n w_i}, \quad (10)$$

Portanto, nós deste nível calculam o grau de verdade normalizado de cada regra. Assim, o número de nós neste nível é igual ao número de nós do nível anterior, que corresponde ao número n de regras.

- **Nível 4:** Todo nó i neste nível é um nó adaptativo computanto a seguinte função:

$$O_i^4 = \overline{w_i}(\rho_i w_i + \sigma_i), \quad (11)$$

onde $\overline{w_i}$ e w_i correspondem as saídas dos i-ésimos nós das níveis 2 e 3, respectivamente. O conjunto $\{\rho_i, \sigma_i\}$ representa os parametros do nó i. Parametros neste nível são referidos como **parâmetros dos consequentes**. Este nível efetua o processo de *defuzzyficação* . A parcela $\rho_i w_i + \sigma_i$ corresponde ao valor da função inversa da função de pertinencia do termo linguístico do consequente da regra em w_i, conforme especificado na *defuzzyficação* de Tsukamoto (equações (3) e (4)).

- **Nível 5:** Todo nó neste nível é um nó não-adaptativo, que computa a saída da rede adaptativa através da soma dos sinais que chegam até ele, ou seja,

$$Output = O_i^5 = x_1 + x2 + \ldots + x_m \quad (12)$$

onde x_k denota o k-ésimo elemento do vetor de entrada do nó, que recebe os sinais do nível imediatamente anterior. O número de nós neste nível determina o tamanho do vetor de saída da rede.

Um algoritmo de aprendizado híbrido[4], que consiste na combinação do método do decaimento do gradiente e da estimativa dos mínimos quadrados, é aplicado na determinação e ajuste dos parâmetros dos nós adaptativos da rede. A partir da arquitetura Anfis equivalente a inferência de Tsukamoto, pode-se constatar que dado os valores dos parametros do nível 1 e os vetores de entrada, a saída da rede pode ser expressa como combinação linear dos parâmetros do nível 4. Precisamente a saída *output* da rede adaptativa na Figura 2 pode ser reescrita como:

$$\begin{aligned} output &= \frac{w_1}{w_1+w_2}\mu_{C_1}^{-1}(w_1) + \frac{w_2}{w_1+w_2}\mu_{C_2}^{-1}(w_2) \\ &= \overline{w_1}\mu_{C_1}^{-1}(w_1) + \overline{w_2}\mu_{C_2}^{-1}(w_2) \\ &= (\overline{w_1}w_1)\rho_1 + (\overline{w_2}w_2)\rho_2 + (\overline{w_1})\sigma_1 + (\overline{w_2})\sigma_2 \end{aligned} \quad (13)$$

Isto é possível devido a restrição de que as funções de pertinência dos conjuntos difusos associados à conclusão das regras devam ser lineares(equação (1)) o que implica que a sua função inversa, usada pelo processo de *defuzzyficação*, será também linear (equação (4)).

Em virtude da saída da rede Anfis ser uma combinação linear dos parametros do nível 4 (equação (13)), a estimativa dos mínimos quadrados pode ser utilizada para determinar o valor desses parametros durante o passo *forward* em que P vetores de treinamento são apresentados à rede e os sinais propagam-se na direção da saída da rede, de forma a obter-se o seguinte sistema de equações:

$$AX = B \quad (14)$$

onde X é um vetor desconhecido cujos elementos são os parâmetros dos nós do nível 4. Sendo o número total de parâmetros do nível 4 igual a M, então as dimensões de A, X e B são $P \times M$, $M \times 1$ e $P \times 1$, respectivamente. Desde que P, o número de pares entrada-saída do conjunto de treinamento, é normalmente maior que M, isto é um sistema com mais equações do que variáveis e geralmente não existe nenhuma solução exata para a equação (14). Ao invés disto, a estimativa dos mínimos quadrados de X, que minimiza o erro quadrático $||AX - B||^2$, é calculada através do algoritmo do filtro de Kalman[8]:

$$\left.\begin{aligned} X_{i+1} &= X_i + S_{i+1}a_{i+1}^T(b_{i+1} - a_{i+1}X_i) \\ S_{i+1} &= S_i - \frac{S_i a_{i+1}^T a_{i+1} S_i}{1+a_{i+1}S_i a_{i+1}^T}, i = 0, 1, \ldots, P-1 \end{aligned}\right\} \quad (15)$$

onde a_i é a i-ésima linha da matriz A definida na equação (14), b_i é o i-ésimo elemento de B, S_i é frequentemente denominado **matriz de covariância.** e a estimativa dos mínimos quadrados de X é igual a X_P. As codições iniciais para equação (15)

são X_0 igual a matriz nula de dimensão $M \times 1$ e $S_0 = \gamma I$, onde γ é um número positivo grande e I é a matriz identidade de dimensão $M \times M$. Quando a rede Anfis apresenta múltiplas saídas, a equação (15) continua aplicável exceto que b_i é a i-ésima linha da matriz B.

O método do decaimento do gradiente é usado para ajustar os parametros do nível 1, não alterados pela estimativa dos mínimos quadrados, sendo aplicado no passo *backward* em que as taxas de erros, propagam-se do extremo de saída da rede para o extremo de entrada da rede. Considerando P vetores de treinamento, pode-se definir a **medida do erro** para o p-ésimo vetor, com $1 \leq p \leq P$, do conjunto de treinamento como sendo a soma dos quadrados dos erros:

$$E_p = \sum_{m=1}^{\#(5)} (T_{m,p} - O^5_{m,p})^2 \qquad (16)$$

onde $\#(5)$ representa o número de nós do nível 5; $T_{m,p}$ é o m-ésimo componente do p-ésimo vetor de saída desejado; e $O^5_{m,p}$ é o m-ésimo componente do vetor saída que foi realmente obtido a partir da apresentação do p-ésimo vetor de entrada à rede Anfis equivalente a inferência de Tsukamoto. Logo, a medida do erro total é $E = \sum_{p=1}^{P} E_p$.

Para desenvolver um procedimento de aprendizagem que implementa o decaimento do gradiente de E sobre o espaço dos parametros, primeiro é necessário calcular a **taxa de erro** $\frac{\partial E_p}{\partial O}$ para cada nó. A taxa de erro para a saída do nó i do nível 5 pode ser calculado prontamente da equação (16):

$$\frac{\partial E_p}{\partial O^5_{i,p}} = -2(T_{i,p} - O^5_{i,p}) \qquad (17)$$

Analogamente, pode ser derivada a taxa de erro para o i-ésimo nó de um nível interno k como combinação linear das taxas de erros dos nós dos níveis subsequentes. Considerando α como um parâmetro da rede adaptativa em questão, tem-se:

$$\frac{\partial E_p}{\partial \alpha} = \sum_{O^* \in S} \frac{\partial E_p}{\partial O^*} \frac{\partial O^*}{\partial \alpha} \qquad (18)$$

onde S é o conjunto de nós cuja saída depende de α. Logo a derivada do erro total E com relação a α é

$$\frac{\partial E}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \alpha} \qquad (19)$$

Logo, as fórmulas para atualização dos parametros α_i, β_i e γ_i do i-ésimo nó do nível 1 são definidas por:

$$\left.\begin{array}{rcl} \Delta\alpha_i &=& -\eta \frac{\partial E}{\partial \alpha_i} \\ \Delta\beta_i &=& -\eta \frac{\partial E}{\partial \beta_i} \\ \Delta\gamma_i &=& -\eta \frac{\partial E}{\partial \gamma_i} \end{array}\right\} \qquad (20)$$

onde η representa a taxa de aprendizado.

A derivação da rede Anfis equivalente ao sistema de inferencia difusa de Takagi e Sugeno[10] pode ser encontrada em [4].

4 Aplicação como Aproximador de Função

Uma das características primordiais de um sistema de inferência difusa é ser um aproximador funcional, de forma que possa simular adequadamente o comportamento do sistema a ser modelado. Objetivando verificar tal característica, a rede Anfis equivalente a inferência difusa de Tsukamoto foi utilizada para modelar a seguinte função não-linear:

$$y = \frac{sin(x)}{x} \qquad (21)$$

Foram amostrados 33 pares de dados de treinamento e 32 pares de teste a partir da equação acima uniformente espaçados no intervalo [-16, 16] e [-15.5, 15.5], respectivamente. A estrutura da rede Anfis usada apresentava cinco nós nas níveis 1, 2, 3 e 4; e mais um nó no nível 5. Esta rede equivale a um sistema de inferência difusa de Tsukamoto com 5 regras, uma variável de entrada e uma de saída. Cada nó do nível 1 possui tres parâmetros(α, β, γ), e cada nó do nível 4 apresenta dois parâmetros(ρ, σ). Portanto o número total de parâmetros da rede é igual a 25. Esses parâmetros são ajustados pelo algoritmo de aprendizado híbrido. Considerando a ausência de especialista ou conhecimento disponível a priori, os parâmetros das premissas são inicialmente estabelecidos, de forma que as suas funções de pertinência associadas sejam igualmente espaçadas ao longo do intervalo de operação de cada variável de entrada. Valores iniciais para os parâmetros dos consequentes não são necessários, uma vez que os valores destes parametros serão apropriadamente determinados pela estimativa dos mínimos quadrados. Na fase de treinamento após 100 ciclos aplicando o algoritmo de aprendizado híbrido, o erro quadrado médio estabilizou em 0.008. Na Figura 3, são exibidas as curvas obtidas pelos valores de teste através da equação (21), da saída da rede Anfis, e do erro absoluto entre esses valores. Pode-se constatar que, de fato, a rede Anfis equivalente ao sistema de inferencia difusa de Tsukamoto apresenta uma ótima aproximação da função não-linear definida pela equação (21).

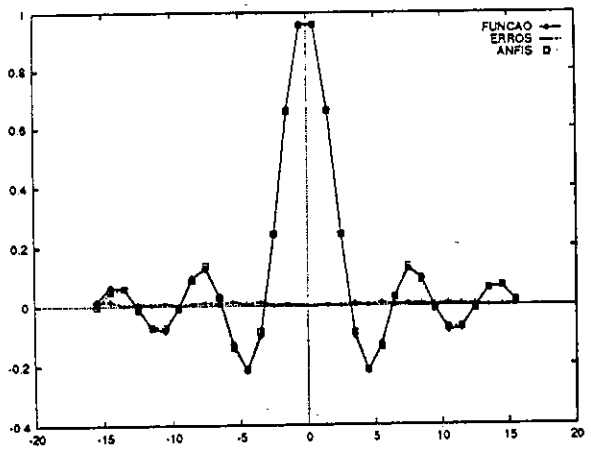


Figura 3: Resultados obtidos pela rede Anfis equivalente a inferencia difusa de Tsukamoto

5 Conclusão

A modelagem do sistema de inferência difusa de Tsukamoto através de uma rede Anfis equivalente visa enriquecer este importante e bastante usado tipo de sistema de inferência difusa com capacidade de aprendizado e refinamento de regras difusas, de forma que objetivos prescritos possam ser atingidos da melhor maneira possível.

Durante o processo de aprendizado, as funções de pertinencia podem sofrer alterações que não preservam características linguisticamente desejáveis, tais como suficiente sobreposição das funções de pertinência e cobertura total do intervalo do domínio, criando dificuldades para a interpretação linguística das regras obtidas. Neste caso, a solução consiste em elevar o número dos parametros das premissas e dos consequentes através de uma maior quantidade de nós na estrutura da rede(ou seja, um maior número de regras e funções de pertinencia), para prover a rede com maior grau de liberdade. Isto faz com que os parâmetros não precisem sofrer mudanças consideráveis, de modo que as funções de pertinencia obtidas preservam as características que facilitam sua interpretação linguística.

Em suma, a rede Anfis equivalente ao Sistema de inferencia difusa de Tsukamoto pode automaticamente derivar as regras IF-THEN difusas se um especialista não é disponível, ou refinar as regras IF-THEN difusas obtidas de um especialista. No último caso, um processo de aprendizado mais rápido é esperado, pois o conjunto de regras e das funções de pertinência estabelecidas pelo especialista se encontra mais próximo da configuração ideal.

Referências

[1] C. Lee. *Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part 1.* IEEE Trans. on Systems, Man, and Cybernetics. Vol. 20, No. 5, 1990

[2] C. Lee. *Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part 2.* IEEE Trans. on Systems, Man, and Cybernetics. Vol. 20, No. 5, 1990.

[3] C. Lin, C. G. Lee. *Neural-Network-based Fuzzy Logic Control and Decision System.* IEEE Trans. on Computers, Vol. 40, No. 12, 1991.

[4] J. R. Jang. *ANFIS: Adaptative-Network-Based Fuzzy Inference System* . IEEE Trans. on Systems, Man, and Cybernetics. May, 1993.

[5] B. Kosko. *Neural Networks and Fuzzy Systems.* Prentice-Hall, USA, 1992.

[6] E. H. Mandani. *Applications of Fuzzy Algorithms for Control of Simple Dynamical Plants.* IEE Proc. Vol. 121, No. 12, 1974.

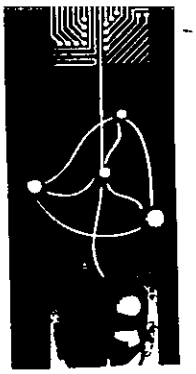
[7] W. S. McCulloch, W. H. Pitts. *A logical calculus of the ideas immanent in nervous activity.* Bull Math Biophys. Volume 5, 1943, p115-133.

[8] C. Goodwin, K. Sin. *Adaptative Filtering Prediction and Control.* Prentice-Hall, N. J., 1984.

[9] D. E. Rumelhart, G. E. Hinton, R. J. Williams. *Learning Internal Representations by Error Propagation.* Parallel Distributed Processing, MIT Press, 1986, p318-362

[10] T. Takagi, M. Sugeno. *Derivation of Fuzzy Control Rules from Human Operator's Control Actions.* Proc. of the IFAC Symp. on Fuzzy Information, Knowledge Representation and Decision Analysis, p55-60, 1993.

[11] Y. Tsukamoto. *An Approach to Fuzzy Reasoning Method.* Advances in Fuzzy Set Theory and Applications, North-Holland, Amsterdam, 1979.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

Requisitos de um Ambiente para Simulação de Redes Neurais Artificiais [1]

Patrícia Duarte de Lima Machado
Edson Costa de Barros Carvalho Filho

UFPE - Universidade Federal de Pernambuco
Departamento de Informática
Cx 7851, 50.732-970, Recife, PE, Brasil
Telefone: (081)271-8430, Fax: (081) 271-4925
Internet: {pdlm, ecdbcf}@di.ufpe.br

Sumário

Este artigo apresenta alguns requisitos a serem considerados na construção de ambientes para simulação de redes neurais artificiais. O objetivo é identificar, com um alto nível de abstração, os objetivos, as características e as restrições, tal que possam ser analisadas por futuros usuários do ambiente e que sirvam como base para o processo de desenvolvimento do software. Para elaboração dos requisitos, foram realizadas entrevistas com pesquisadores de redes neurais, iniciantes e especialistas, e foi estudado um grupo representativo dos simuladores atuais. Os requisitos definidos estão sendo usados no desenvolvimento do ambiente EASY.

Abstract

This paper shows some requirements to be considered in the construction of artificial neural networks simulation environments. The goal is identify, at a high abstraction level, objectives, features and restrictions, which may be analysed by users and may be used at the software development process. To elaborate the requirements, novice and expert neural network researchers were consulted and a representative group of actual simulators were studied. The requirements defined are been used at the environment EASY development.

[1] Este trabalho foi desenvolvido com o apoio do CNPq.

1 Introdução

Simulação em computador é uma opção comumente adotata por projetistas de redes neurais, visando especificar arquiteturas e verificar os seus comportamentos. A falta de uma notação formal que permita a realização de provas formais para verificação de propriedades é o maior fator que contribui para que a simulação continue sendo uma atividade dominante. Algumas notações já foram propostas [San94, Fie94], mas não possuem semantica formal. Mesmo que tal notação exista, sua aplicação, apesar de aparentemente mais confiável, torna-se difícil a medida que aumenta o número de componentes envolvidos no processo. Por esta razão, a simulação é vista como um importante aliado ao desenvolvimento das pesquisas em redes neurais [HBG90, HGR92].

Existem tres tipos básicos de implementação de redes neurais [Sim90]: *implementações em software*, executadas em um computador convencional, *implementações em hardware*, qualquer implementação eletronica com único propósito de processar um ou mais modelos de redes neurais, e *implementações óticas*, construídas utilizando componentes óticos.

Implementações em hardware dedicado, utilizando técnicas VLSI, visam dar mais velocidade a execução das redes. Uma desvantagem é que a capacidade de modificar o modelo fica comprometida. Uma outra desvantagem é que o tempo para desenvolver um sistema deste tipo é tipicamente da ordem de meses. Contudo, seu uso é adequado às situações nas quais o modelo a ser usado estiver completamente definido e testado [HGR92].

Implementações em software perdem em velocidade com relação às implementações em hardware e óticas, mas ganham em flexibilidade para modificações e expansões do modelo escolhido. Normalmente, são feitas utilizando uma linguagem de programação de propósitos gerais como C, C++, Smalltalk, C++ Concorrente, entre outras. Apesar de flexível, apresenta como desvantagem o fato de que um software tem de ser desenvolvido sempre que um paradigma tiver que ser simulado para uma dada aplicação. Se apenas a etapa de programação

for considerada no processo de desenvolvimento do software, então o produto resultante não deverá ter a qualidade e confiabilidade necessária, visto que nenhuma metodologia, técnica ou método é adotado para evitar a presença de definições ambíguas, inconsistentes e incompletas. Deverá também ser de difícil manutenção. A possibilidade de ser reutilizado por outros projetistas é praticamente nula, devido a dificuldade de se entender um código em uma linguagem de programação convencional. Outro problema com este tipo de simulação é que o projetista perde um valioso tempo com a implementação do software. Este tempo poderia ser utilizado em outros estudos.

Um simulador de rede neural pode ser definido como um pacote de software criado com o propósito específico de reduzir o tempo e o esforço envolvidos na solução de um problema usando redes neurais [Mur94]. Sistemas deste tipo devem permitir que pesquisadores possam testar novas idéias com um esforço mínimo e prover o processamento necessário a investigação de modelos complexos. Os simuladores de redes neurais podem ser classificados em tres tipos: *programa demonstrativo*, cujo objetivo é ilustrar um paradigma de rede especfico e seu comportamento; *simulador de propósito específico*, que possibilita a definição de um número variado de aplicações usando um conjunto particular de paradigmas de rede; e *ambiente de simulação*, que dá suporte a uma larga classe de paradigmas de rede bem como a definição e investigação de novos.

A utilização de um simulador oferece muitas vantagens sobre a construção de um *software* específico. Geralmente, permite que o usuário especifique a rede com um alto nível de abstração e de uma forma mais compacta, ao contrário do que ocorreria se o modelo tivesse que ser construído usando uma linguagem de programação. Muitos pesquisadores de redes neurais não são especialistas em computadores. Portanto, podem não dispor de tempo ou da habilidade necessária a construção de software.

O grande problema dos simuladores atuais é que estes não dão suporte a realização de um experimento completo, ou seja, não cobrem todas as etapas do processo de simulação de uma rede neural. Além disso, na grande maioria dos casos, possuem uma interface de difícil utilização e, portanto, consomem quase o mesmo tempo que o pesquisador levaria para construir seus próprios programas. Por, este motivo, na prática, estes simuladores são mais utilizados por iniciantes na área, os quais realizam pequenos experimentos a fim de aprender e explorar os diversos paradigmas.

EASY (*An [E]nvironment for [A]rtificial Neural [SY]stems Simulation*) [MFMG94] é um ambiente de simulação cujo principal objetivo é dar suporte a realização de um experimento completo usando redes neurais artificiais. Especificações Formais e Orientação a Objetos foram adotadas em seu processo de desenvolvimento visando assegurar qualidade, confiabilidade e reusabilidade ao produto final. A especificação formal utilizada pode ser encontrada em [GM93, MMFG94].

As seções seguintes apresentam um modelo para o processo de simulação de redes neurais e a definição de requisitos. Por fim, são dadas as conclusões sobre o trabalho realizado.

2 Um Modelo para o Processo de Simulação de Redes Neurais

Assim como existem modelos para o processo de desenvolvimento de software os quais especificam as atividades a serem seguidas para a geração de um produto, existe também a necessidade da definição de um modelo para o processo de simulação de uma rede neural que seja largamente aceito e que possa ser comumente adotado. Esta necessidade deve-se ao fato de que a computação neural é também uma abordagem para o processamento de informações destinada a aplicação em problemas para os quais algoritmos e regras não são conhecidos ou são, porém os softwares para implementá-los são caros, consomem muito tempo ou são inconvenientes para desenvolver [HN90].

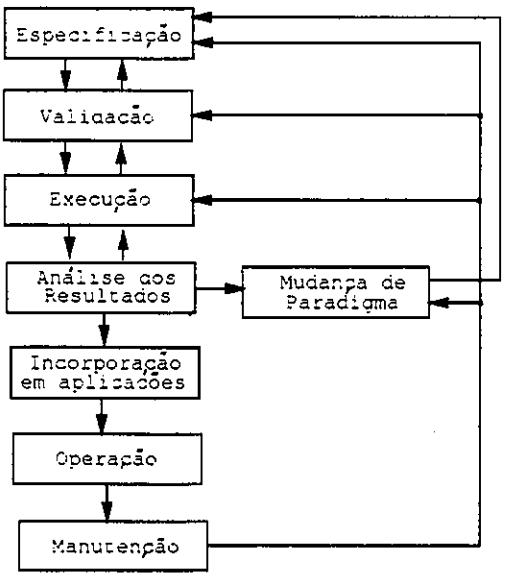


Figura 1: Um modelo para o ciclo de vida de aplicações usando redes neurais.

A concepção de um ambiente para simulação de redes neurais deve levar em consideração a existencia de um modelo para o processo, a fim de

poder fornecer o suporte necessário a realização do mesmo. Somente a adoção de um modelo pode conduzir a completude do ambiente, ou seja, o fornecimento de todo o suporte necessário. Caso nenhum modelo seja adotado, os usuários poderão ter que utilizar outras ferramentas a fim de complementar as etapas não cobertas pelo software de simulação escolhido. Na prática, muitos simuladores neurais se concentram em prover eficiência em apenas algumas das etapas do processo [Mur94].

[Mur94] sugere um conjunto de etapas que, na maioria dos casos, devem ser seguidas para a simulação de uma rede neural. Tais etapas são as seguintes:

1. **Especificação**. Consiste na tradução de um problema em uma representação de rede neural. Isto implica em escolher um conjunto de padrões a ser apresentado a rede, o subconjunto que melhor representa o domínio da aplicação, o paradigma a ser adotado, e a definição da arquitetura da rede, isto é, a quantidade, distribuição e conectividade dos neurônios.

2. **Validação**. Nesta etapa, é feita uma validação das escolhas efetuadas na etapa 1, tais como: a arquitetura da rede, o subconjunto escolhido como representativo do domínio da aplicação, entre outros.

3. **Execução da Simulação**. Dependendo do paradigma escolhido, consiste na aplicação dos algoritmos apropriados para efetuar a aprendizagem da rede.

4. **Análise dos Resultados**. Nesta etapa, são analisadas as saídas geradas pela rede após o processo de aprendizagem, o nível de generalização e a capacidade de abstração e outros parâmetros de interesse de acordo com o paradigma escolhido.

5. **Escolha de um outro Paradigma**. O paradigma que está sendo utilizado pode ser extendido para incluir características adicionais que melhor se adequem a solução do problema em questão ou mesmo um novo paradigma pode ser escolhido. Esta etapa ocorre com muita freqüência, principalmente se um novo problema estiver sendo tratado. A execução desta etapa implica num retorno obrigatório à 1.

6. **Incorporação em Aplicações Reais**. Consiste na geração de programas de aplicação ou esquemas para implementação em hardware.

Estas etapas podem ser interpretadas como a tentativa de definição de um modelo para o processo de simulação de redes neurais. A figura 1 apresenta um modelo mais completo para o ciclo de vida de aplicações usando redes neurais, onde é incluída também as fases de uso ou operação e manutenção da rede. Assim como no modelo *waterfall* [Som89], que é um modelo de desenvolvimento de software, o retorno a uma etapa anterior é previsto. A manutenção é feita para efetivar mundanças no domínio da aplicação ou corrigir erros.

3 Definição de Requisitos

A definição de requisitos é uma descrição em linguagem natural dos requisitos de um software. Consiste em uma descrição dos conceitos do sistema e a justificativa para cada decisão tomada. Existem dois tipos básicos de requisitos: requisitos funcionais e não-funcionais. Os requisitos funcionais definem a forma como o software deve operar e os requisitos não-funcionais definem os seus objetivos, características e restrições.

Após a realização de um estudo sobre alguns simuladores e de entrevistas com pesquisadores da área, foram definidos alguns dos requisitos básicos que um ambiente de simulação de redes neurais deve possuir a fim de que possa ser amplamente aceito e utilizado. Vale ressaltar que a completude destes requisitos fica comprometida devido ao grande número e a grande diversidade de usuários. Segundo [Som89] é muito difícil formular uma especificação definitiva para sistemas de software de grande porte. Assim, a medida que novos usuários forem sendo consultados estes requisitos poderão mudar e novos requisitos poderão ser identificados.

Neste trabalho, são apresentados apenas os requisitos não-funcionais por falta de espaço. Os requisitos funcionais podem ser encontrados em [Mac94]. Os requisitos não-funcionais identificados são os seguintes:

1. **Múltiplos Paradigmas**. Devem ser fornecidos pelo ambiente diferentes paradigmas de redes neurais, incluindo diferentes tipos de neurônios, arquiteturas e algoritmos de aprendizagem, para que o usuário possa escolher o modelo mais adequado a solução do seu problema. Este é um dos objetivos de projeto do SESAME [LSTW92].

2. **Diferentes Aplicações**. Diferentes tipos de padrões devem poder ser utilizados pelo sistema a fim de possibilitar a definição de diversos tipos de aplicações. Este é um outro objetivo do SESAME.

3. **Flexibilidade para Incorporação de Novos Paradigmas**. Devido a constante evolução da teoria de redes neurais, é imprescindível que um ambiente de simulação ofereça

facilidades para a incorporação de novas arquiteturas, modelos de neuronio e algoritmos de aprendizagem. Esta característica está intimamente relacionada com a evolutibilidade do software. A reutilização de definições é de grande importância para que pesquisadores tornem-se aptos a testar novas idéias com rapidez e com um mínimo de esforço. A orientação a objetos tem sido utilizada como maneira preferencial para a modelagem de redes neurais, pois promove a modularidade e a reusabilidade [HBG90, LSTW92, FAT93].

4. **Execução em Hardware Paralelo**. Redes neurais possuem um carater inerentemente paralelo e é crescente o número de implementações que tiram proveito da execução paralela para diminuir o tempo de simulação das redes. O simulador parsimANNs [MWGH92] possibilita execução em hardware paralelo.

5. *Interfaces* **de fácil entendimento e utilização**. Devem permitir que o usuário possa utilizar o ambiente sem ter que se preocupar com os seus detalhes operacionais, ou ter que perder muito tempo lendo manuais exaustivos com comandos para interação.

6. *Interfaces* **Consistentes**. Uma interface consistente é aquela em que quaisquer dois comandos, opções de menu ou botões que possuam uma identificação com a mesma sintaxe devem produzir um comportamento análogo. Esta é uma característica encontrada no ambiente de programação SmallTalk [Sys90]. Uma vez tendo aprendido os comandos de uma dada ferramenta do ambiente, em qualquer outra ferramenta onde forem encontrados comandos com a mesma sintaxe, estes serão facilmente compreendidos pelo usuário.

7. *Interfaces* **Reusáveis**. A reusabilidade é um fator primordial para viabilizar a expansão do sistema. Por interfaces reusáveis podemos entender, por exemplo, que uma interface para dar suporte a uma atividade de um dado paradigma deve ser projetada, tal que, um novo paradigma incorporado no ambiente reutilizando as definições do primeiro possa também reutilizá-la na definição de sua própria. Reutilização de interfaces é discutido em [ES92].

8. **Interfaces com diferentes visões de usuário**. De acordo com o tipo de usuário, um maior ou menor número de opções deve ser apresentado. Por exemplo, para um iniciante, apenas algumas opções devem ser fornecidas, devido ao seu pouco conhecimento. Porém para um usuário especialista tanto na área de redes neurais quanto na própria utilização do ambiente, um maior número de opções deve ser apresentado.

9. **Especificação de arquiteturas selecionando objetos da interface**. O objetivo é facilitar o trabalho do projetista de uma rede neural, evitando que este tenha que construir programas ou aprender uma seqüência de comandos para especificar as arquiteturas com as quais deseja trabalhar, reduzindo, significativamente, o tempo dispendido para esta tarefa. Na interface ANNview [MWGH92], uma parte das definições são efetuadas usando *mouse* com operações, *point, clic e drag* em *icons* gráficos. Assim, o usuário não precisa ser especialista em programação para definir novos paradigmas ou testar os pré-existentes. Os simuladores RCS [GLM88] e UCLA-SFINX [MS92] adotam a linguagem de programação *C* para a especificação de arquiteturas.

10. **Especificação de novas funções de ativação/aprendizagem utilizando uma linguagem declarativa com sintaxe e semântica formais**. Linguagens procedurais induzem a especificação de "como" deve ser implementado, aumentando o trabalho do projetista. Linguagens declarativas possuem um alto nível de abstração, possibilitando a descrição do "que" deve ser implementado. Linguagens formais são baseadas em matemática e, por este motivo, evitam definições ambígüas e inconsistentes. A maioria dos simuladores utilizam linguagens procedurais. No simulador NeuroGraph [Wil93], funções são especificadas usando expressões matemáticas.

11. **Execução** *foreground* × *background*. A execução da rede em *foreground* é importante, pois possibilita que o usuário possa acompanhar o processo. Porém, uma vez aprovados todos os parâmetros a serem utilizados, a execução em *background* se torna mais adequada, visto que, para redes grandes, o tempo de execução é, normalmente grande, e, portanto a execução em *background* libera o usuário para executar outras atividades. Os simuladores SNNS [ZMH+93] e SESAME [LSTW92] são exemplos de simuladores que fornecem as duas opções.

12. **Interação durante a execução em** *foreground*. O usuário quando executando uma simulação em *foreground*, deve ser capaz de acompanhar e controlar a evolução do processo, tal que possa interrompê-lo para alterar alguns parametros. Este é o caso do SESAME [LSTW92].

13. **Salvar as informações de status**. Todas as definições dos usuários devem poder ser guar-

das em um arquivo para que em uma posterior sessão de trabalho, estas possam ser utilizadas sem que o usuário precise especificá-las novamente. No Aspirin [Lei92], o treinamento pode ser interrompido e o estado da rede pode ser salvo para ser carregado posteriormente, dando continuidade ao processo.

14. **Auto-inicialização**. Todas as definições necessárias a execução de uma atividade devem ser automaticamente inicializadas pelo sistema para evitar que resultados inconsistentes sejam gerados. Isto é necessário caso o usuário forneça uma especificação incompleta. O SESAME [LSTW92] trata este problema.

15. *Help on-line* **como um hipertexto**. Um *Help* com estrutura não-linear, ou seja, na forma de um hipertexto [SFAM92], deve estar disponível para qualquer ferramenta do ambiente. Ele deve ser sensível ao contexto da ferramenta a partir da qual for invocado, mas, por outro lado, deve permitir que por navegação, o usuário tenha acesso a qualquer informação sobre qualquer ferramenta do sistema. Entre as informações que devem estar disponíveis, temos a semântica dos botões e *menus* das ferramentas, um guia com uma possível sequencia de passos a serem seguidos para concretização de uma tarefa, descrição teórica das funções executadas pela ferramenta (por exemplo, a especificação matemática de um algoritmo de aprendizagem) e a teoria completa sobre o paradigma escolhido. Estas duas últimas informações seriam de grande importância para iniciantes na área de redes neurais.

16. **Anotações** Todas as ferramentas do ambiente devem permitir que anotações particulares sejam feitas pelo projetista, para que este possa registrar, a cada etapa, comentários próprios sobre as decisões que forem sendo tomadas. Estas anotações devem ser guardadas juntamente com as informações de *status* da ferramenta e as opções feitas pelo projetista. Um dos problemas do SESAME [LSTW92] é que este não armazena os comentários do projetista juntamente com as especificações de experimentos.

17. **Customização do Ambiente**. As opções de visualização gráfica e textual fornecidas em todas as ferramentas devem ser opcionais, para que o usuário possa escolher a que mais lhe convier e para que não haja um sobrecarga cognitiva com um grande número de informações sendo apresentadas ao mesmo tempo. O UCLASFINX [MS92] permite que o usuário defina seu ambiente de simulação.

4 Conclusões

O principal objetivo deste trabalho é identificar, com um alto nível de abstração, os requisitos de um ambiente para simulação de redes neurais, tal que seja amplamente aceito e utilizado. A importância da aquisição destes requisitos é justificada, quando se leva em consideração o grande número e diversidade de usuários deste tipo de sistema. Muitos estudos têm mostrado que erros encontrados em fases mais avançadas do processo de desenvolvimento de software são mais caros para correção do que os detectados no início do processo. A definição dos requisitos permite que os objetivos sejam estabelecidos e investiga as necessidades dos usuários e o domínio da aplicação.

Os requisitos apresentados neste trabalho podem ser aplicados no processo de desenvolvimento de qualquer ambiente para simulação de redes neurais artificiais. Em particular, está sendo utilizado no desenvolvimento do ambiente EASY como base para a especificação formal de suas propriedades.

Referências

[ES92] Raimund K. Ege and Christian Stary. Designing maintainable, reusable interfaces. *IEEE Software*, pages 24–32, november 1992.

[FAT93] L. Fuentes, J. F. Aldana, and J. M. Troya. Urano: An object-oriented artificial neural network simulation tool. Em J. Mira, J. Cabestany, and A. Prieto, editors, *New Trends in Neural Computation: International Workshop on Artificial Neural Networks - IWANN*, pages 364–369. Sitges, Spain, june 1993.

[Fie94] E. Fiesler. Neural network classification and formalization. To be published in "Computer Standards & Interfaces, volume 16, special issue on Neural Network Standards, John Fulcher, editor. North-Holland, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, 1994, ISSN 0920-5489, 1994.

[GLM88] Nigel H. Goddard, Kenton J. Lynne, and Toby Mintz. Rochester connectionist simulator. Relatório Técnico 233, University of Rochester. Computer Science Department, Rochester, New York, 14627, march 1988.

[GM93] H. M. Gomes and P. D. L. Machado. Especificando Redes Neurais Artificiais

em MooZ. Relatório técnico, Universidade Federal de Pernambuco, Departamento de Informática. Recife - PE. 1993.

[HBG90] G. L. Heileman, H. K. Brown, and M. Georgiopoulos. Simulation of artificial neural network models using an object-oriented software paradigm. Em *Proceedings of the International Joint Conference on Neural Networks*, pages II–133–II–136, Washington, DC, 1990.

[HGR92] G. L. Heileman, M. Georgiopoulos, and W. D. Roome. A general framework for concurrent simulation of neural network models. *IEEE Transactions on Software Engineering*, 18(7):551–562, july 1992.

[HN90] Robert Hecht-Nielsen. *Neurocomputing*. Addison-Wesley Publishing Company, Inc. 1990.

[Lei92] R. R. Leighton. *The Aspirin/ MIGRAINES Neural Network Software - User's Manual*. MITRE Corporation, 1992.

[LSTW92] A. Linden, Th Sudbrak, Ch Tietz, and F. Weber. An object-oriented framework for the simulation of neural nets. Em C. L. Giles, S. J. Hanson, and J. D. Cowan, editors, *Advances in Neural Information Processing Systems 5*. CA: Morgan Kaufmann Publishers, San Mateo, 1992.

[Mac94] Patrícia D. L. Machado. Proposta de um ambiente para modelagem, simulação e análise de redes neurais artificiais. Tese de Mestrado, Universidade Federal de Pernambuco, Departamento de Informática, Recife - PE, Brasil. 1994.

[MFMG94] P. D. L. Machado, E. C. D. B. Carvalho Filho, S. R. L. Meira, and H. M. Gomes. EASY - an [E]nvironment for [A]rtificial neural [SY]stems simulation. To be published at Fourth Irish Neural Network Conference - INNC'94, Dublin, Ireland, 1994.

[MMFG94] P. D. L. Machado, S. R. L. Meira, E. C. D. B. Carvalho Filho, and H. M. Gomes. Especificando redes neurais artificiais em MooZ. To be published at XX Conferencia Latino Americana de Informatica - PANEL'94, Mexico, 1994.

[MS92] Edmond Mesrobian and Josef Skrzypek. A software environment for studying computacional neural systems. *IEEE Transactions on Software Engineering*, 18(7):575–589, july 1992.

[Mur94] Jacob M. J. Murre. Neurosimulators. To be published in the "HandBook of Brain Research and Neural Networks", M. A. Arbib, MIT Press, 1994.

[MWGH92] Harley R. Myler, Arthur R. Weeks, Randall K. Gillis, and Gary W. Hall. Object-oriented neural simulation tools for a hypercube parallel machine. *Neurocomputing*, 4(5):235–248, 1992.

[San94] D. A. Santos. MOdelo Formal de Especificação Universal para Redes Neurais - MOFEU. Tese de Mestrado, Universidade Federal de Pernambuco, Departamento de Informática, Recife - PE, Brasil, 1994.

[SFAM92] A. C. Salgado, D. Fonseca, E. S. Albuquerque, and S. R. L. Meira. *Sistemas Hipermídia: Hipertexto e Banco de Dados*. VIII Escola de Computação, Gramado, RS, 1992.

[Sim90] Patrick K. Simpson. *Artificial Neural Systems: Foundations, Paradigms, Aplications and Implementations*. Pergamon Press, Inc. 1990.

[Som89] Ian Sommerville. *Software Engineering*. Addison-Wesley Publishing Company, 3 edition, 1989.

[Sys90] ParcPlace Systems. *Objectworks.\ Smalltalk - User's Guide*. 1550 Plymouth Street, Mountain View, California 94043, 1990.

[Wil93] Peter Wilke. Simulation of neural networks in a distributed computing environment using neurograph. Em J. Mira, J. Cabestany, and A. Prieto, editors, *New Trends in Neural Computation: International Workshop on Artificial Neural Networks - IWANN*, pages 394–398, Sitges, Spain, june 1993.

[ZMH+93] A. Zell, N. Mache, R. Hübner, G. Mamier, M. Vogt, K. Herrmann, M. Schmalzl, T. Sommer, A. Hatzigeorgiou, S. Döring, and D. Posselt. SNNS: Sttugart Neural Network Simulator - User Manual. Relatório Técnico 3, University of Stuttgart, Institute for Parallel and Distributed High Performance Systems, Breitwiesenstr. 20-22, 70565 Stuttgart, Fed. Rep. of Germany, 1993.

1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

NEUWORK - AMBIENTE DE DESENVOLVIMENTO DE APLICAÇÕES UTILIZANDO REDES NEURAIS

G. Lambert-Torres,
L.E. Borges da Silva,
E.C. Saturno,
A.P. Alves da Silva

Escola Federal de Engenharia de Itajubá
Grupo de Inteligência Artificial

Abstract - Este artigo apresenta um pacote computacional para o treinamento "off-line" e "on-line" de redes neurais, bem como um simulador de malha de controle de velocidade de um acionamento de motor de corrente contínua.

1. INTRODUÇÃO

NEUWORK é um pacote de ferramentas com interface totalmente amigável, orientado para o desenvolvimento de aplicações de redes neurais na área de controle.

O pacote é constituído por três programas: NEUWORK I, NEUWORK II e NEUCONTROL, para computadores IBM PC ou compatíveis, sob ambiente DOS 2.0 ou versão superior. Necessita-se um mínimo de 300 Kbytes no disco para os arquivos executáveis e de dados. O pacote utiliza alocação dinâmica de memória, o que possibilita um melhor aproveitamento da memória RAM.

O pacote cria e manipula arquivos com extensão:
- .neu -----> estrutura de rede.
- .par -----> conjunto de pesos.
- .dat -----> conjunto de padrões para treinamento "off-line".
- .eqd -----> estrutura e parâmetros de equações as diferenças.

2. NEUWORK I

NEUWORK I é um programa destinado ao treinamento tipo "off-line" de redes neurais do tipo multicamadas, permitindo, entre outras funções, a criação, treinamento e teste das redes. O treinamento é realizado através do algoritmo "back-propagation" utilizando várias técnicas de treinamento rápido.

A tela principal do programa, figura 1, mostra a estrutura, em termos de camadas e número de neurônios, da rede.

Através da opção TREIN. do menu principal podem ser ajustadas as faixas de normalização (figura 2) e os parâmetros do treinamento (figura 3), entre eles, o valor das taxas de treinamento, o erro global mínimo desejado e o número máximo de iterações. Estão disponíveis para o treinamento, duas opções de aceleração do algoritmo, através de taxas de treinamento variável e através de ganhos auxiliares.

O conjunto de padrões de treinamento pode ser selecionado para treinamento, construído e visualizado através da opção VET-TR.

O treinamento pode ser acompanhado através das saídas da rede, das saídas desejadas e dos erros parciais e globais, atualizados em intervalos de tempo definidos pelo usuário (figura 4).

Após o treinamento, a rede pode ser testada utilizando a opção RESULTADO com entradas através de janelas (figura 5).

O programa permite ainda a criação de arquivos de erros para geração de gráficos (figura 6), usados na análise do comportamento do treinamento e comparação entre as diversas técnicas de aceleração. A saída pode ser tanto na tela como na impressora.

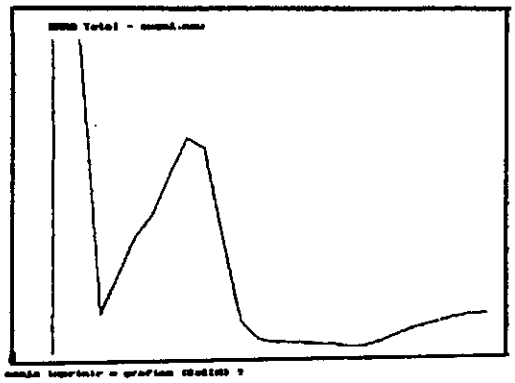


Figura 6

3. NEUWORK II

Simulações de treinamento em aplicações do tipo "on-line" são realizadas neste programa. O NEUWORK II é um ambiente de simples utilização para simular as funções de identificação e treinamento de controladores utilizando redes neurais. O sistema a ser identificado e controlado é introduzido através de sua equação as diferenças (figura 7), a qual pode ser carregada de um arquivo (.eqd) ou criada dentro do próprio ambiente.

A opção REFER. permite a construção de um sinal de referência constituído por degraus ou rampas ou funções do tipo senoidal.

O programa possui uma interface gráfica para cada um das opções de treinamento. A identificação (figura 8) permite a utilização de várias configurações para a rede, assim como o acompanhamento do procedimento, não só através dos valores dos sinais mais importantes a cada iteração, mas também, através de gráficos (figura 9) e da variação dos pesos (figura 10). O treinamento do controlador (figura 11) possui os mesmos recursos mostrados no procedimento de identificação.

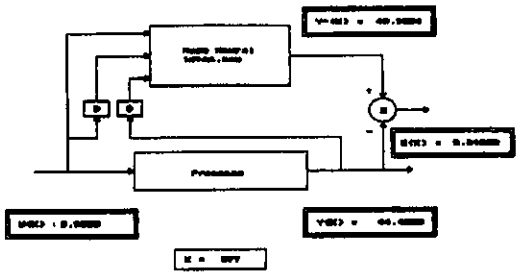


Figura 8

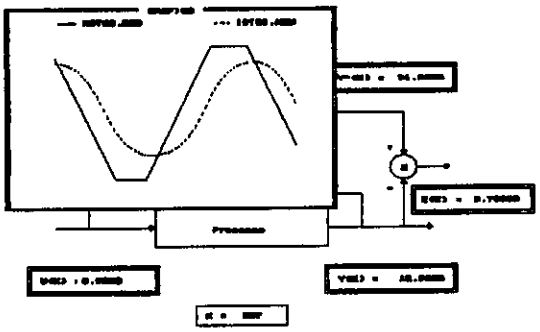


Figura 9

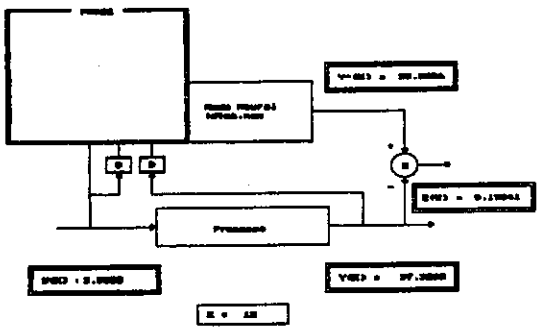


Figura 10

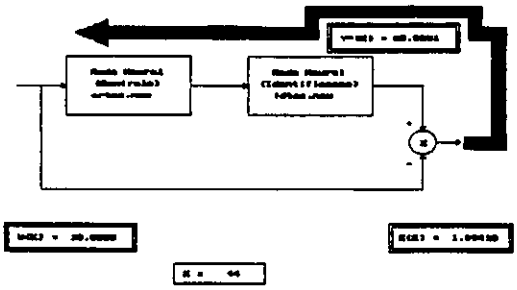


Figura 11

Os resultados podem ser conferidos através de simulação em gráficos tanto na tela (figura 12) como na impressora, identificando os sinais envolvidos na parte superior do gráfico.

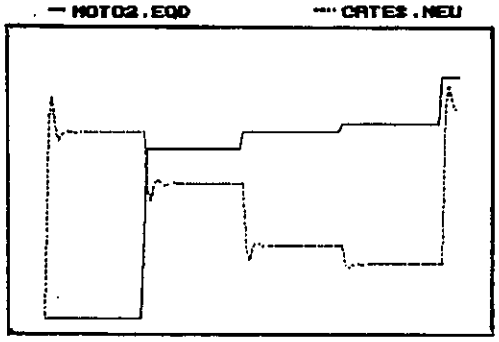


Figura 12

4. NEUCONTROL

Finalmente, o programa NEUCONTROL é um simulador da malha de controle de velocidade de um acionamento de motor CC, cujos parâmetros podem ser introduzidos e modificados facilmente (figura 13).

O programa inclui a escolha do tipo de controlador (figura 14) entre vários tipos conhecidos, entre técnicas de resposta rápida "deadbeat response" ou inclusive a utilização de um sistema de controle neural, com redes treinadas dentro do NEUWORK II.

Para análise de desempenho, o programa tem a capacidade de introduzir modificações nos parâmetros do acionamento ou ruído no sinal de realimentação do transdutor, simulando situações que podem ser encontradas no ambiente de trabalho. Na figura 15, por exemplo, pode ser especificada uma variação no coeficiente de fricção do conjunto motor carga, simulando variações na carga, determinando o instante e o tempo de variação. Pode-se também, especificar o nível de ruído a ser introduzido na realimentação.

O programa possui também, saídas gráficas das simulações, tanto na tela como na impressora, da resposta da malha a uma referência especificada (figura 16).

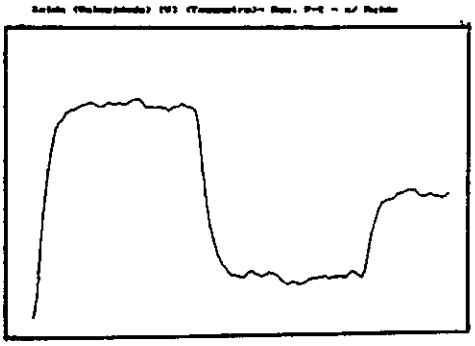


Figura 16

5. CONCLUSÃO

Este artigo apresentou um ambiente de desenvolvimento de redes neurais. Diversas aplicações tem sido realizadas com este pacote, principalmente, em aplicações industriais.

6. AGRADECIMENTOS

Este trabalho foi possível devido a ajuda do CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico e da FAPEMIG - Fundação de Amparo à Pesquisa de Minas Gerais.

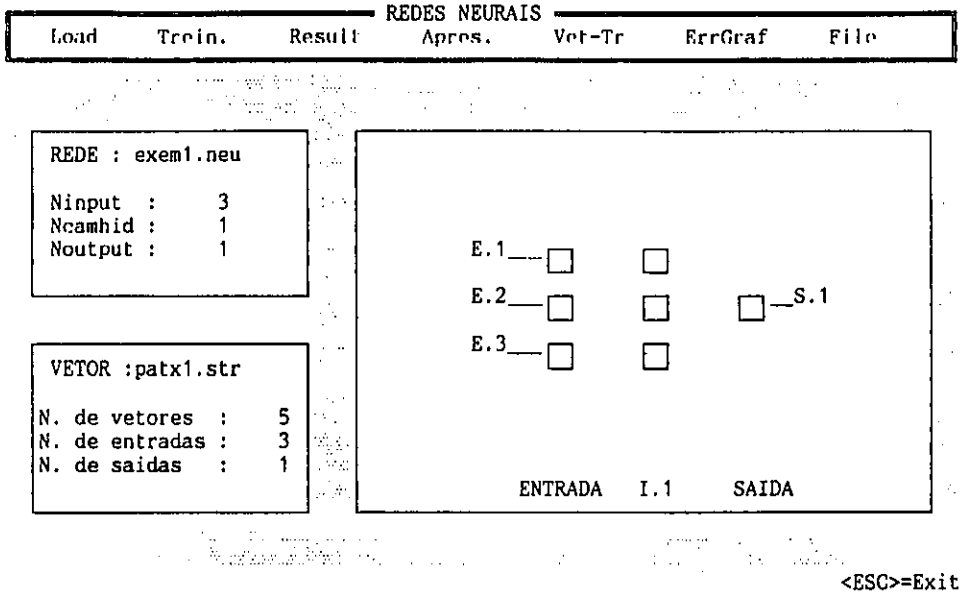


Figura 1

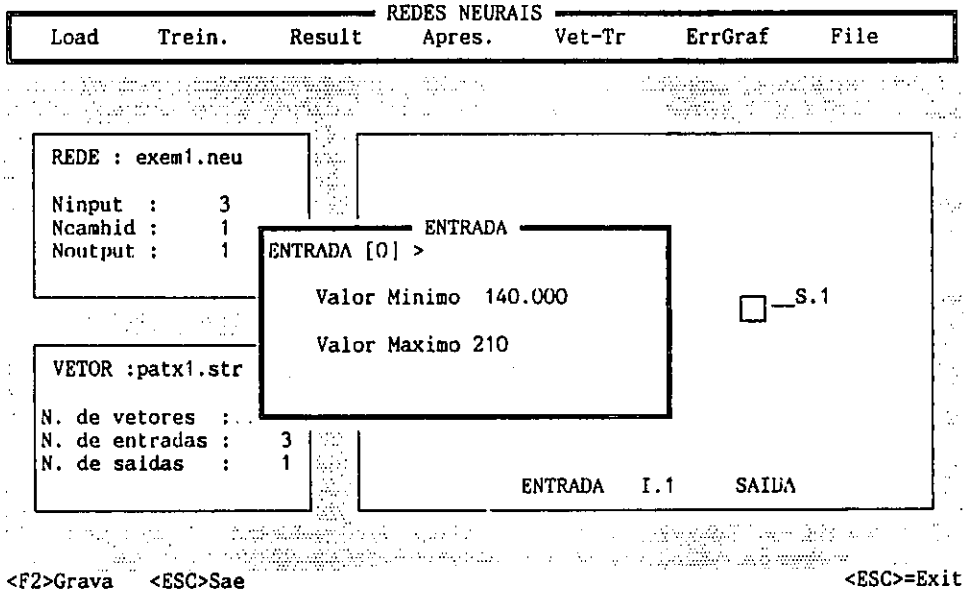


Figura 2

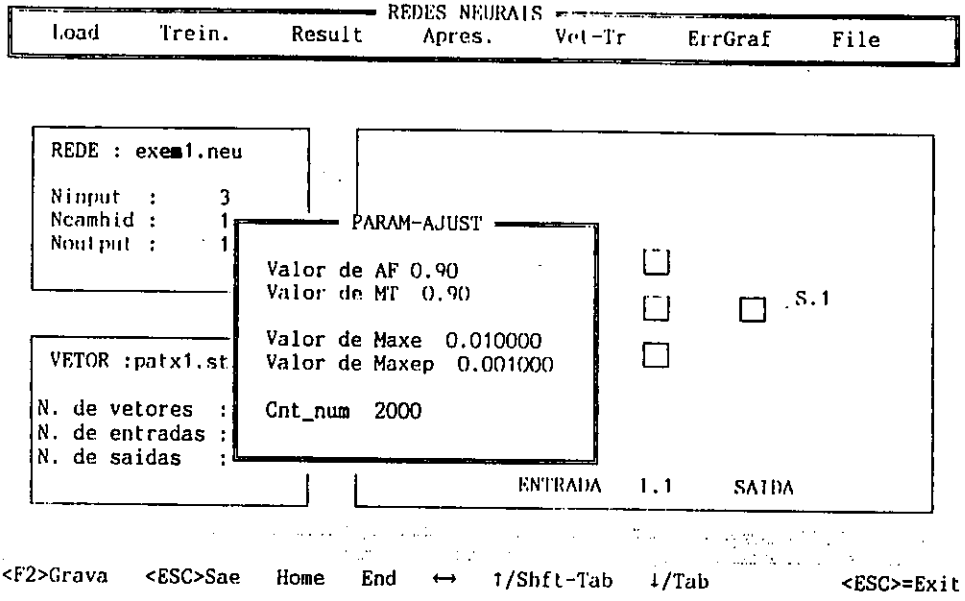


Figura 3

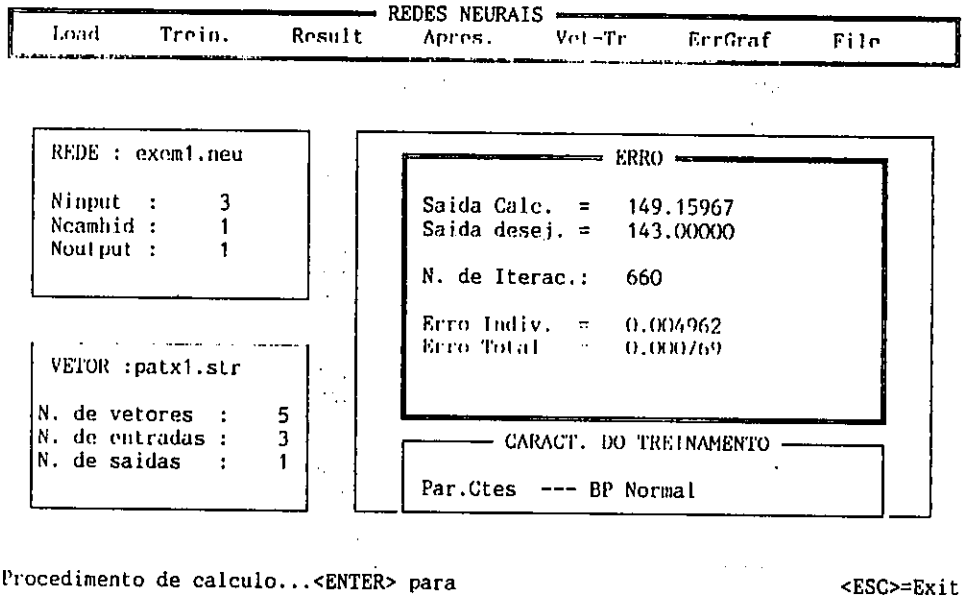


Figura 4

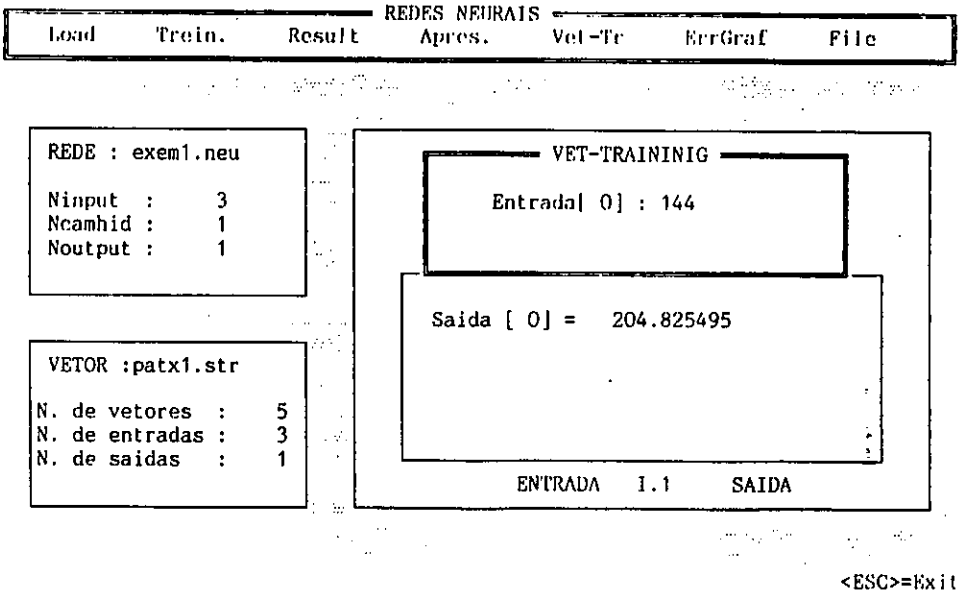


Figura 5

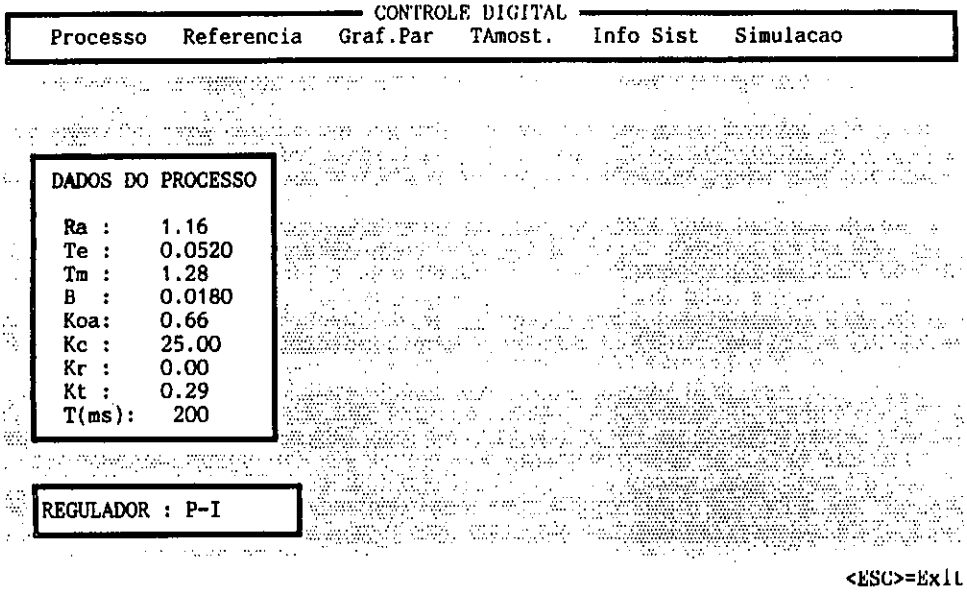


Figura 13

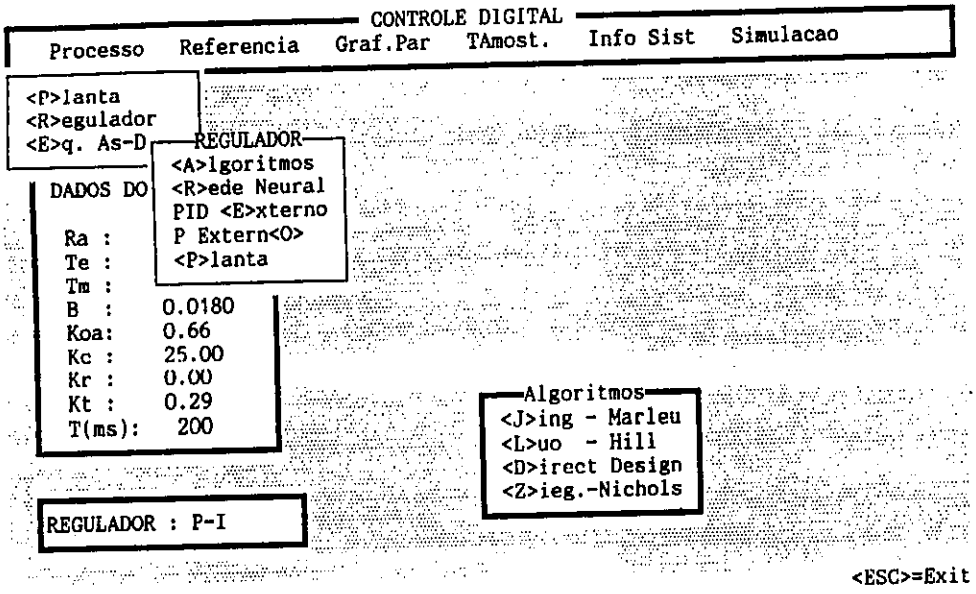


Figura 14

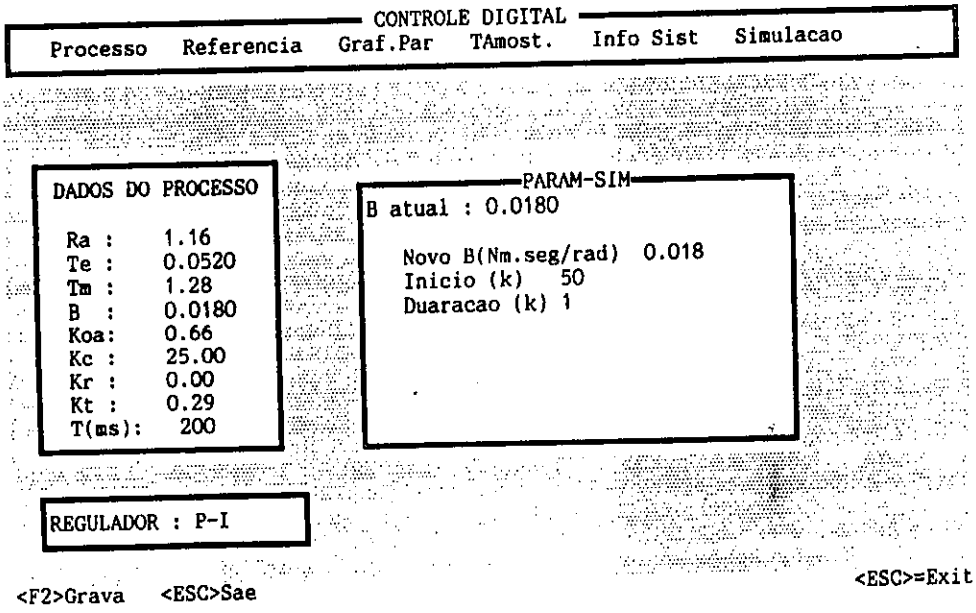
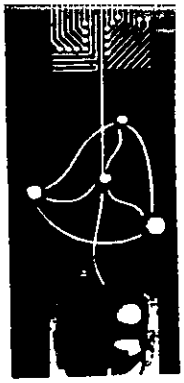


Figura 15



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajuba
Itajuba, 24 a 27 de outubro de 1994

HENNS: Heterogeneous Environment Neural Network Simulator *

Wagner Meira Junior[†] André Luiz de Senna [‡] Márcio Luiz Bunte de Carvalho [§]

Departamento de Ciência da Computação – UFMG
Caixa Postal 702 – Belo Horizonte – MG – Brazil – CEP 30.161-970
Tel: +55.31.443-4088 Fax: +55.31.443-4352

Key-Words: Neural Networks, Simulation, Heterogeneous and parallel processing

Abstract

The simulation of artificial neural networks (ANN) is a hard task due to two reasons: the complexity of ANN specification and the computational cost associated. The first problem has been addressed by some simulators by allowing specification of ANNs. Parallel processing has been recognized as a solution for the second problem. This work presents a neural network simulation strategy that runs in a heterogeneous environment, that combines different types of parallel processors and servers. This strategy is implemented in the HENNS, an environment that allows the user to describe and simulate various neural network models in parallel and heterogeneous environments. The results show that the use of heterogeneity is better than the use of parallel architectures alone.

1 Introduction

Artificial neural networks are models based in the human brain, which works via the interaction of a great number of appropriately interconnected computationally simple nodes, which mimics the behavior that is believed of the neurons. A neural network stores its information on the weights associated to the connections between pairs of nodes and in the activation level of each node. Using simple algorithms, neural systems are reliable and capable of dealing with a great amount of information. However, their simulation shows to be a hard task mainly due to two reasons: the functional complexity of the whole and its high computational cost.

The implementation of the models has been facilitated by the development of various general-use neural network simulators. These tools allows the simulation of neural networks providing model's flexibility and functioning abstraction.

On the other hand, more efficient neural network simulators alternatives are found in the literature, for example, using hardware implementation and/or parallelism. Hardware implementations are the fastest, but has smaller flexibility; parallel implementations are less efficient but more flexible, a more suitable approach for scientific investigation.

This work presents HENNS, a software neural network simulator intended to run on an heterogeneous and parallel environments. This system provides tools that enables the user define the neural network model to be simulated and various parameters that specify the simulation itself. From this specification, the simulator generates an appropriate C source-code, compiles and links it building an executable code that performs the simulation.

This paper has six sections: this is the first, an introduction; the second reviews some previous work in sequential and parallel neural network simulation; section three describes the computing enviroment used; the fourth and fifth sections presents HENNS and some results acnieved with its utilization; the conclusions and future work are presented in the last section.

*This work was sponsored by FAPEMIG (proc. TEC 1113/90) and CNPq (proc. 502353/91-0(NV)).

[†]meira@cs.rochester.edu
[‡]senna@dcc.uimg.br
[§]mlbc@dcc.uimg.br

2 Neural Network Simulators

2.1 Sequential Simulators

The sequential neural network simulators in the literature can be divided up into two classes. The first simulators are batch-oriented; a neural model is completely specified and then all the simulation is performed: in this case, the user can not interact with the simulation in any way. PABLO [Perkel, 1976] and BOSS [Wittie, 1978] are examples of these simulators.

The second class is the interactive simulators, which enables the user to modify (and/or visualize graphically) the simulation while it is being performed. In these simulators, the network to be simulated is specified via a description language, which is either compiled or interpreted to perform the simulation. In the case were graphical interface environment is used, primitives that allows changes and verification of the simulation events are commonly provided. In this class we can cite ISCON [Small et al., 1983], GENESIS, Asprin, MIRRORS/II [D'Autrechy et al., 1988], SFINX [Skrzypek and Mesrobian, 1992] and PLANET [Miyata, 1991]. Although they work similarly, these simulators differ from each other in the complexity and flexibility in specifying the neural network to be simulated; we find from analogical to particular model simulators. However, all of them have the common goal of joining abstraction with flexibility of constructing neural networks. A more complete reference on these simulators and their features can be found in [Meira Jr., 1993b].

2.2 Parallel Simulations

The inherent parallelism of neural networks has motivated their simulation on parallel machines. MIMD and SIMD machines are the most popular target architectures. In each of these architectures we find different problems to be addressed, as it will be shown below.

Parallel implementations explore the processing independence of the neural networks nodes, i.e. all nodes that has all its necessary inputs may start its processing phase. The major problem in these simulations is how establish efficient communication among the nodes.

In a MIMD architecture this problem is harder: to achieve efficiency, we must also guarantee load balance amongst the processes and the correct arrival sequence of the messages exchanged between the asynchronous processors. We find several proposals in the literature, which have been implemented in various architectures such Intel iPSC Hypercube, BBN Butterfly, SBC and mainly Transputers. We can conclude that the most important factor in a MIMD implementation is the communication costs, i.e. the interconnection pattern of the network to be simulated is more significative than its local processing.

The use of SIMD architectures in neural network simulation presents new variables. An important one is the architectural restrictions of these machines. Examining related works, where implementations on machines such Zephyr, DAP, CM-2 and Hughes SCAP are described, we can conclude that SIMD implementations are extremely machine dependent, in spite of some general mapping strategy proposed [Salles et al., 1992].

A interesting proposal is given by Nordstron [Nordström and Svensson, 1992], who affirms that the MIMSIMD architectures are the most suitable for neural network simulation. His proposal and the problems experienced during the simulation of neural networks on conventional parallel architectures has motivated us to investigate the use of heterogeneous architectures in the simulating neural networks.

3 The Computing Environment

In order to verify the suitability of the neural networks simulation in distributed and heterogeneous processing environments, we used two groups of resources available at DCC-UFMG: (i) a Sun workstation network with the PVM package: (ii) the Zephyr Wavetracer SIMD machine: they are described below:

PVM: PVM (Parallel Virtual Machine)[Geist et al., 1993] is a software package that allows an heterogeneous network of parallel and serial computers to appear as a single concurrent computational resource. PVM consists of two parts: a daemon process that any user can install on a machine, and a user library that contains

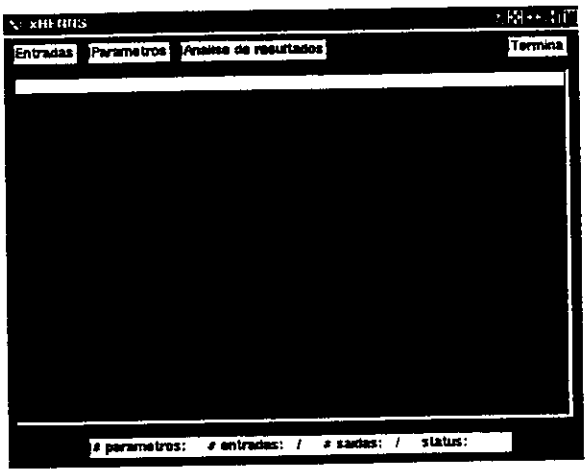


Figure 1: xHENNS initial screen

routines for initiating processes on other machines, for communicating between processes, and changing the configuration of machines.

Zephyr-Wavetracer: the Zephyr Wavetracer (WT) is a 8,192 1-bit processors SIMD machine. Each processor have two kinds of memory: (i) an internal cache memory of 256 bytes; (ii) an external one of 32 kbytes. Another interesting feature is that the number of active processors is not limited to the quantity of physical processors, because it is allowed to create virtual processors. This machine works as a Sun co-processor and there is an ANSI C language extension, the multiC, to access and use it. So, the executable code is executed sequentially in the host (a Sun workstation) and, when a multiC instruction is found, it is performed in the Wavetracer.

4 HENNS

HENNS [Meira Jr., 1993b. Meira Jr., 1993a], developed at DCC-UFMG [1], is a neural network simulator in which the user can define topologies and neural network models arbitrarily. HENNS provide also means of specifying simulation parameters, such as the computing environment: sequential, distributed and vectorized. In the sequential form, all of the network parts are combined in a single executable program. In the distributed and vectorized executions, the network and each of its components that must be simulated by itself corresponds to a PVM process. The vectorized programs uses the Zephyr-Wavetracer primitives. In the distributed manner if operation, each of its sequential processes are implemented as PVM processes.

The development of neural applications using HENNS can be divided into two phases: application and neural network. The application works as a front-end of the network and can be freely modified by the user. The network can be seen as a "black–box" which receives inputs and parameters and returns outputs and parameters. The protocol between application and network is simple and implemented via PVM messages.

Included in the HENNS package there are two examples of application software, the first one, called cHENNS, is a ascii based interface to HENNS generated neural networks; the second, xHENNS has the same functionality of the former with a xWindows interface. The initial screen of the later can be seen in the Figure 1.

The neural network implementation using HENNS, can be divided in three steps: (i) definitions of the neural architecture and its simulation using the description language of HENNS; (ii) run HENNS to read the definitions and generate a C code that performs the simulation; (iii) compile and execute this code.

Following, each of these steps will be described in details:

4.1 Neural Architecture Definition

In this step, the user specifies his simulation using HENNS' description language. This specification is made in two levels:

Structural: in the HENNS, a neural network is defined as an hierarchical structure of three levels: (i) **Network**: represents the neural network as a processing unit for the external tasks and serves as basis of the various cluster's functioning; (ii) **Cluster**: it is an abstraction that join similar nodes, which have the same input and output connections and work similarly. In the back-propagation [Rumelhart et al., 1986] neural network model, for example, each level of the network may be seen as a cluster; (iii) **Node**:

[1] HENNS is available via *anonymous* ftp at dcc.ufmg.br

it's the smallest functional unit of a neural network, the neuron.

To completely define the structure, one needs also to define the connections between the network parts. In HENNS, these connections can be between the network and some cluster or between clusters. It is not permitted intra-clusters connections due to efficiency and parallelization considerations. Because cluster is a abstraction proposed by HENNS, this should not cause any problem.

Functional: the definition of the simulation behavior is made in a procedural way. The neural network (and the other hierarchy parts) may work based on various operation modes, each defined as a list of commands drawn from a set of primitives that deal with various aspects, some of them are:

- Protocol: they implement easy exchange of information between the network hierarchical parts, independently of the architecture and machine under use;
- Control Flow: these are commands that control the execution flow: while e if, that makes the procedural language more powerful and flexible;
- User-defined functions: any function related to the specific internal behavior of each network part in the various operation modes are defined by the user. This definition is made using a target machine language, to access HENNS hierarchical data structures and library functions.

4.2 Source-code Generation

In this step, based on the specification files, various information are generated:

Definition: constants and other network specification definitions;

Initialization: functions that performs network components data and protocol initializations;

Operation: functions that implement the network work;

Compilation: specifications and compilation dependencies of the network and its components, described using the syntax of a Unix program called make;

Simulation: a Unix *shell script* with the steps necessary for the execution: *daemons* instantiation, compilation and the execution of the simulation.

4.3 Source-code Compilation and Execution

In order to compile the generated application, the Unix program make is invoked and everything is performed automatically: the initialization and operation source-codes generated are compiled and linked to the HENNS library. If the simulation is to be performed sequentially, only one executable code is generated. If a network with three clusters should be performed distributed, four executable codes are generated: one for the network and one for each cluster. If there are vectorized processing, the multiC compiler is used and an executable is generated as it is done in the distributed way.

The simulation is the execution of the network code. When the simulation is distributed, the network program initiates the others, performing all the initialization and termination control of the "child" processes.

5 Results

The results presented next are part of an investigation of the most suitable architecture for neural network simulation in the DCC-UFMG hardware environment. In order to perform the measures, a back-propagation neural network was specified and his topological characteristics changed during the various tests.

5.1 Test Environment

The tests presented below were performed in the Sun network of the DCC: the SIMD machine Zephyr-Wavetracer was attached to a Sun Sparc Station 2 also connected to the DCC network. To perform the tests presented below, we used a back-propagation neural network with three levels as described in [Hertz et al., 1991]. Concerning to

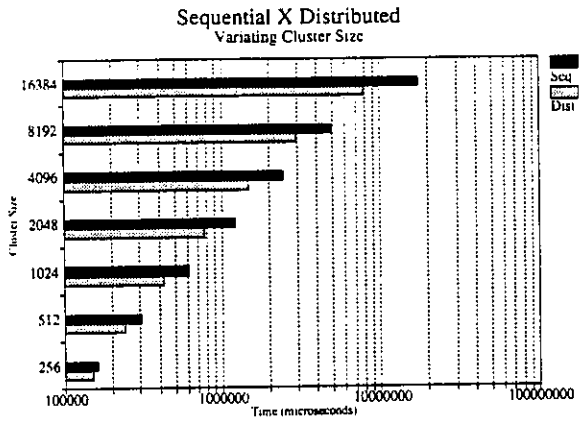


Figure 2: Sequential × Distributed

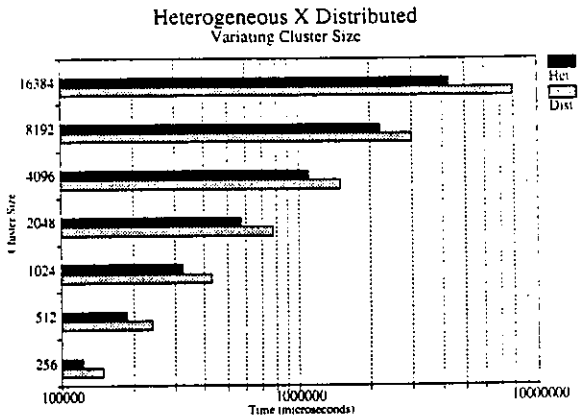


Figure 3: Heterogeneous × Distributed

machine configurations we have used the following:

Sequential: the simulation was performed in only one workstation. It was used two models of SUNs: Sparc Station 2 and Sparc Station SLC;

Distributed: The clusters or parts of clusters are executed in SLC machines, one per machine:

Vectorized: It was performed by two processes in the WT host machine: (i) the vectorized cluster to be simulated: (ii) an executable that performs the remaining parts of the network sequentially. This differentiation was necessary due to the fact that the programing language used in vectorized implementation (multiC) is different from the one used by the sequential;

Heterogeneous(distributed/vectorized): in these simulations, it was used three kinds of executable code-machine pair: (i) the network program executed in a Sparc 2; (ii) the vectorized cluster performed in the WT host; (iii) the other clusters performed in SLC machines. one per machine.

Below, we will present a comparison among the various machine configurations, varying some of the topological parameters of the network simulated. A more complete set of tests can be seen in [Meira Jr., 1993b]. Although, we won't present any result using a pure vectorized configuration. it is described for completeness purpose.

5.2 Distributed × Sequential

In this test we verify the suitability of distributed processing to simulate clusters with various number of nodes. So we create neural networks where the input and output levels have always the same number and variate the length of the hidden level. Both the simulations, sequential and distributed, are performed in Suns SLC.

The plot of Figure 2 shows the results for clusters that varies from 256 to 16384 nodes. The distributed executions were clearly better than the sequential ones. The speedup varies from 10% (for the smallest cluster) to 70% (for the biggest cluster). this can be explained by the overhead of the communication between the processes.

5.3 Distributed × Heterogeneous

After we verified that the distributed and vectorized simulations are better than the sequential ones. we investigates the gain due to the using of both solutions simultaneously. The WT host machine is used as a node in a distributed simulation. configuring an heterogeneous environment.

In the graph of Figure 3 we can see the performance of the heterogeneous and distributed simulation. The heterogeneous simulations was always better with the speedup varying from 20 to 80% with the size of the hidden cluster.

6 Conclusion and Future Work

This works has presented a strategy for simulating neural networks in heterogeneous environments

and an ANN simulator that adopted this strategy, HENNS.

Using the Sun network of DCC-UFMG and the SIMD machine Zephyr-Wavetracer, we verified the suitability of distributed and vectorized simulations in that computing environment, followed by the analysis of our strategy. In this analysis, we noted that the proposed strategy allows performance enhancements proportional to the size of the network simulated. This can be explained by communication costs associated to these implementations. Quantitatively, the distributed simulation time was 80% greater than the heterogeneous time in the best case. If we consider sequential and heterogeneous time, this difference increases up to an order of magnitude.

Future work includes simulation of other ANNs models and the extension of HENNS for different computing environments. The enhancement of the application tools is also a goal, allowing the user not only simulates his neural network, but also visualize and tune the executions on the fly.

References

[D'Autrechy et al., 1988] D'Autrechy, C. L., Reggia, J. A., Sutton, G. C., and Goodall, S. M. (1988). A general-purpose simulation environment for developing connectionist models. *Simulation*, 51(1).

[Geist et al., 1993] Geist, A., Beguelim, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V. (1993). *PVM 3.0 User's Guide and Reference Manual.* ORNL/TM-12187 Oak Ridge National Laboratory.

[Hertz et al., 1991] Hertz, J., Krogh, A., and Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation*, volume 1 of *Computation and neural systems series*. Allan M. Wylde.

[Meira Jr., 1993a] Meira Jr., W. (1993a). Guia do usuário e manual de referência do HENNS. Technical Report RT013/93, DCC-UFMG.

[Meira Jr., 1993b] Meira Jr., W. (1993b). Implementação de redes neuronais em ambientes paralelos. Master's thesis, Departamento de Ciência da Computação - UFMG, Belo Horizonte, MG.

[Miyata, 1991] Miyata, Y. (1991). *A User's Guide to PlaNet Version 5.6.* University of Colorado at Boulder.

[Nordström and Svensson, 1992] Nordström, T. and Svensson, B. (1992). Using and designing massively parallel computers for artificial neural networks. *Journal of Parallel and Distributed Computing*, (14):260 – 285.

[Perkel, 1976] Perkel, D. H. (1976). A computer program for simulating a network of interacting neurons. *Computers and Biomedical Research*, (9):31 – 43.

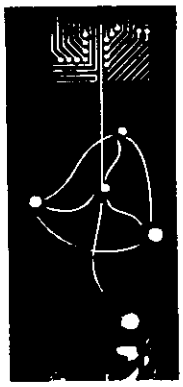
[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, Cambridge, MA.

[Salles et al., 1992] Salles, J., Vieira, M., Meira Jr., W., and Carvalho, M. L. B. (1992). Implementação de redes neuronais em máquinas simd. In *IV Simpósio Brasileiro de Arquiteturas de Computadores e Processamento de Alto Desempenho*, pages 347 – 361.

[Skrzypek and Mesrobian, 1992] Skrzypek, J. and Mesrobian, E. (1992). *UCLA-SFINX - Simulating Structure and Function in Neural Connections.* Machine Perception Lab., UCLA.

[Small et al., 1983] Small, S. L., Shastri, L., Brucks, M. L., Kaufman, S., Cottrell, G. W., and Addanki, S. (1983). Iscon: A network construction aid and simulator for connectionist models. Technical Report TR 109, University of Rochester.

[Wittie, 1978] Wittie, L. D. (1978). Large-scale simulation of brain corticies. *Simulation*, 3(31).

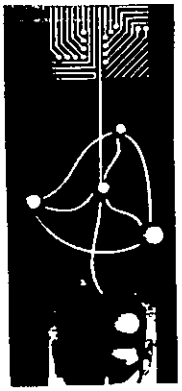


1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

PROCESSAMENTO DE SINAIS I

Anotações



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

Reconhecimento Off-line de Assinaturas Utilizando MLP-Backpropagation e Momentum: um Estudo Comparativo[1]

Herman Martins Gomes
Edson Costa de Barros Carvalho Filho

UFPE - Universidade Federal de Pernambuco
Departamento de Informática
CP 7851, 50.732-970, Recife, PE, Brasil
Fone: (081)271-8430, Fax: (081) 271-4925
Internet: {hmg, ecdbcf}@di.ufpe.br

Abstract

One of the most problems with off-line signature recognition is the drastical reduction of useful information due to the fact that all dynamic features are reduced to a single statical image. A great variety of promising techniques, that have been established as apropriated to general tasks of pattern recognition, may be used in off-line signature recognition to better the performance, like Neural Networks and Momentum. The main goal of this paper is to discuss and to compare Neural Networks (represented by MLP-Backpropagation paradigm) and Moment-based techniques (represented by Standard Moments) in off-line signature recognition. We start with a brief introduction to the off-line signature recognition problem. Afterwards, we give a description of the investigated techniques. At the end, a signature database, the design of experiments and results are presented, and conclusions related.

1 Introdução

O reconhecimento automático de manuscritos representa uma nova tecnologia para facilitar a interface homem-máquina e para integrar melhor a utilização do computador na sociedade [12]. Um dos pontos de interesse na análise de assinaturas (e manuscritos em geral) é a possibilidade de reconhecimento através do processamento de características próprias do autor que são depositadas, mesmo que inconscientemente, sobre o papel quando o mesmo escreve. Inúmeros sistemas estratégicos de informação requerem algum tipo de processamento automático de escrita, como aqueles encontrados em bancos para verificação da autenticidade de documentos. Como exemplo prático desta necessidade, a verificação, ainda manual, das assinaturas de cheques são um ponto crítico do sistema bancário brasileiro, que movimentou, só no primeiro semestre de 1992 cerca de 1,7 bilhão de cheques e outros papéis de compensação [9].

Há duas abordagens utilizadas no reconhecimento de assinaturas: off-line (ou estática) e on-line (ou dinâmica) [8, 12]. O reconhecimento off-line utiliza apenas características extraídas a partir de imagens digitalizadas, como a distribuição de *pixels* ou as variações de tonalidade. Já o reconhecimento on-line considera características dinamicas, tais como, a aceleração da caneta, a pressão desempenhada, a seqüência de movimentos primitivos, dentre outras. Um dos grandes problemas que dificutam o reconhecimento off-line é a redução drástica do número de características observáveis, ao se considerar as assinaturas na forma de imagens estáticas. Portanto, é fundamental a utilização de ténicas efetivas para superar este tipo de problema, tais como Redes Neurais e Momentum.

Vários modelos de redes neurais [5, 13] têm sido propostos , investigados e utilizados em aplicações práticas, das quais se destacam os problemas de reconhecimento de padrões. Destes modelos, MLP Backpropagation [11] foi um dos que adiquiriu grande popularidade, devido a sua simplicidade e poder computacional. Evidentemente, outros modelos também compartilham de propriedades semelhantes, mas, foi escolhido MLP-Backpropagation por este ser mais amplamente conhecido e utilizado. A adequabilidade de Redes Neurais aos problemas de reconhecimento de padrões tem sido confirmada a cada dia. Em particular, o Departamento de Informática da UFPE possui várias teses de mestrado, recentemente defendidas, e publicações na área, que ratificam esta afirmativa [2, 7, 14, 3, 4, 1]. Redes

[1] Desenvolvido com apoio financeiro do CNPq

Neurais podem ser utilizadas tanto para a extração de características quanto para a classificação de imagens de assinaturas. No processo de extração de características, uma rede neural é capaz de fornecer uma representação compacta das imagens. Redes Neurais também são bastante adequadas ao processo de classificação, uma vez que não requerem conhecimento *a priori* das propriedades estatísticas das classes, ao contrário de outras técnicas da teoria de decisão, como os classificadores bayesianos, por exemplo.

Momentum tem se apresentado como uma técnica robusta para a decomposição de uma imagem de forma arbitrária em um conjunto finito de características, uma vez que é baseada na aplicação direta de transformações lineares, não havendo a necessidade de heurísticas específicas da aplicação. Em Estatística, Momentum é utilizado para caracterizar a distribuição de variáveis randômicas, e em Mecânica, serve para caracterizar corpos em função de sua distribuição espacial de massa. Considerando que uma imagem pode ser tratada como uma função de distribuição de densidade bi-dimensional, é possível aplicar Momentum para a análise de imagens, através da extração de propriedades que têm analogia em Estatística e Mecanica.

Considerando as dificuldades com o reconhecimento off-line de assinaturas, procura-se, através deste artigo, discutir a aplicação destas duas técnicas promissoras para extração de características e reconhecimento de imagens, descrever alguns experimentos de reconhecimento em que as técnicas são combinadas em diferentes configurações, apresentar os resultados obtidos, e tecer comentários e conclusões com base nestes resultados. Nas seções seguintes, apresentamos brevemente o modelo MLP-Backpropagation e a técnica baseada em Momentum utilizada.

2 MLP-Backpropagation

Multi Layer Perceptron-Backpropagation (MLP-BP) é um modelo de rede neural artificial que utiliza neurônios do tipo McCulloch-Pitts organizados em múltiplas camadas, possuindo um algoritmo de aprendizagem supervisionada denominado Back-Propagation [11], que baseia-se na correção do erro entre as saídas desejadas e as produzidas pela rede. Nos neurônios deste modelo, usualmente, é utilizada a função sigmóide (equação 1):

$$f(z) = 1/(1 + e^{-z}) \qquad (1)$$

onde $z = \sum_j x_{ij} w_{ij} + \Theta_i$, que corresponde a soma ponderada das entradas pelos pesos mais um *bias*, que é similar a um *threshold* (limiar) no neuronio McCulloch-Pitts.

A arquitetura de uma rede MLP-BP é constituída por duas ou mais camadas de neuronios. A primeira camada é denominada camada de entrada e serve para distribuir os dados através da rede. As camadas seguintes são chamadas de camadas escondidas, e a última camada, de saída, responde com os valores computados pela rede (fig 1).

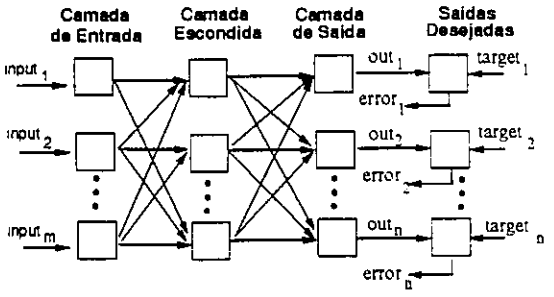


Figura 1: Arquitetura de uma rede MLP-BP

Toda informação codificada numa rede MLP se concentra nos pesos das conexões entre os neurônios, os quais podem ser vistos como parametros para as combinações e composições das funções de ativação (ou de transferencia) dos neurônios. O processo de aprendizagem consiste no ajuste destes pesos de forma que todas as saídas produzidas pela rede representem um erro mínimo com respeito a um fator de vigilância. A capacidade de uma rede MLP aprender uma dada função está diretamente relacionada com a quantidade de neurônios nas camadas escondidas da rede.

Inúmeros sistemas para reconhecimento de padrões baseados no modelo MLP-Backpropagation tem sido propostos e investigados, como por exemplo o que foi definido em [12], para reconhecimento off-line de manuscritos. Várias outras aplicações podem ser encontradas em [5], [13] e [15].

3 Momentum

A fórmula geral para momentum bi-dimensional m_{pq} de ordem $p+q$, de uma função de distribuição de densidade, $f(x, y)$, é dada pela equação 2:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx\, dy \qquad (2)$$

Um conjunto de momentum de ordem n é formado por elementos do tipo m_{pq}, tais que $p + q \leq n$. Para uma imagem discretizada $g(x, y)$ de dimensões $M \times N$, a fórmula para momentum bi-dimensional é dada pela equação 3:

$$m_{pq} = \sum_{y=0}^{M-1} \sum_{x=0}^{N-1} x^p y^q \; g(x, y) \qquad (3)$$

Valores de momentum de baixa ordem representam propriedades geométricas fundamentais de uma distribuição ou de um corpo. Por exemplo, no caso de imagens com função de distribuição binária, o momentum de ordem 0, $\{m_{00}\}$, representa a área total do objeto. Os valores de momentum de ordem 1, $\{m_{01}, m_{10}\}$, podem ser utilizados para calcular o centro de massa $(\bar{x}, \bar{y})$, da imagem do objeto da seguinte forma:

$$\bar{x} = m_{10}/m_{00}, \qquad \bar{y} = m_{01}/m_{00} \qquad (4)$$

Se o objeto for posicionado de forma que seu centro de massa conicida com a origem do sistema de coordenadas ($\bar{x} = 0$ *e* $\bar{y} = 0$), então cada valor de momentum computado para aquele objeto é referido como momentum central, designado por μ_{pq}. Momentum de segunda ordem, $\{m_{02}, m_{11}, m_{20}\}$, conhecido como momentum de inércia, pode ser utilizado para determinar os eixos principais do objeto, sobre os quais há um momentum de segunda ordem mínimo e outro máximo (eixo principais maior e menor, respectivamente). A orientação do eixo principal mais próximo ao eixo x pode ser calculada pela equação 5:

$$\theta = \frac{1}{2} \, tan^{-1}\left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right) \qquad (5)$$

A representação de toda informação contida em um segmento de imagem requer uma quantidade de valores de momentum potencialmente infinita. Em termos práticos, a utilização de Momentum para o reconhecimento de imagens requer a seleção de um subconjunto de valores que contenha informação suficiente para caracterizar unicamente cada imagem. Um dos fatores que diferencia as técnicas baseadas em Momentum é o método para derivar valores invariantes a partir dos valores de momentum calculados sobre a imagem. As principais técnicas baseadas em Momentum que estão atualmente sendo pesquisadas incluem: Invariantes de Momentum, Momentum Rotacional, Momentum Ortogonal, Momentum Complexo e Momentum Padrão [10]. Há duas formas de se obter invariancia: através de invariantes algébricos ou através de normalização de caracteristicas. Em [4], demonstrou-se, experimentalmente, que técnicas baseadas em normalização de caracteristicas tendem a ser mais adequadas ao processamento de imagens de assinaturas do que invariantes algébricos.

Será utilizado Momentum Padrão, uma vez que representa uma técnica não baseada em invariantes algébricos, mas sim em um critério de normalização para escala, posição e orientação. Primeiramente, um conjunto de momentum $\{m_{pq}\}$, de uma ordem desejada, é computado de uma dada imagem, utilizando a equação 3. A partir de então, uma série operações de normalização quanto a escala, posição e orientação é aplicada sobre o conjunto inicial de momentum, para derivar um conjunto padrão de momentum, $\{M_{pq}\}$. Todas as operações atuam apenas no domínio do momentum, não havendo necessidade de manipulação dos pixels da imagem, e combinam os valores de momentum originais de mesma ordem ou menor. Uma descrição detalhada das operações podem ser encontradas no trabalho anterior [4].

Em [10] são relacionados vários trabalhos que tratam da utilização de Momentum em aplicações como compactação de imagens, reconhecimento de caracteres ópticos, objetos sólidos, imagens aérias, e compressão de imagens, dentre outras.

4 Base de Dados de Assinaturas

Para a realização dos experimentos, foi utilizada uma base de dados de assinaturas que está sendo desenvolvida no Departamento de Informática da UFPE. Atualmente, a base de assinaturas conta apenas com amostras verdadeiras de uma comunidade restrita. A intenção foi a de avaliar preliminarmente a adequabilidade de diferentes técnicas no processo mais geral (e mais simples) de reconhecimento de assinaturas, que trata da separação das classes apenas, não considerando, ainda, o problema mais complexo de reconhecimento (classificação) que é a verificação de assinaturas falsificadas. Em sua versão final, a base de dados deverá possuir um número significativo de assinaturas verdadeiras e correspondentes falsificações profissionais (adquiridas em agencias bancárias [9]), além de englobar as culturas gráficas[2] de diferentes regiões do Brasil.

Na versão atual da base de dados, as assinaturas foram coletadas em formulários especiais de 16 × 24cm, com capacidade para 20 amostras. Cada formulário foi utilizado para representar as amostras de uma classe de indivíduo. Um total de 7 classes, com 20 assinaturas cada, foram coletadas entre membros do DI/UFPE. As imagens dos formulários foram digitalizadas em um scanner de mesa, utilizando a resolução de 150dpi e 256 tons de cinza. A combinação de técnicas para equalização de histograma e filtros digitais foi utilizada para melhorar a qualidade das imagens. Em seguida, foi aplicado um algoritmo heurístico (baseado no layout dos formulários) para segmentar as assinaturas em

[2] Entende-se por cultura gráfica o estilo utilizado na escrita das assinaturas, como o tipo das letras, a presença de sinais não alfabéticos etc

retina de dimensões iguais a 350 × 90 pontos. Depois de segmentadas, as imagens foram binarizadas segundo um threshold que permitiu filtrar ruídos sem haver deteriorização da imagem. Com o intuito de permitir avaliar a robustez das técnicas de reconhecimento, não houve normalização das imagens quanto a orientação e escala. Na figura 2 são apresentadas algumas assinaturas pertencentes à base de dados construída.

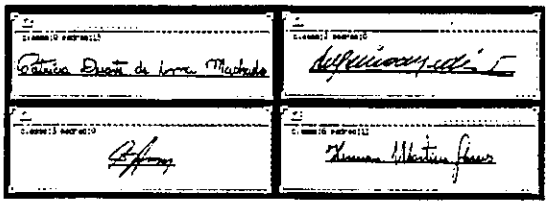


Figura 2: Assinaturas digitalizadas

5 Experimentos

Para cada uma das classes C_j ($j = 0, \ldots, 6$) da base de dados, foram construídos aleatoriamente dois conjuntos de assinaturas, um para treinamento dos classificadores (L_j) e outro para os testes de reconhecimento (T_j). Definiu-se, *a priori*, que ambos os conjuntos teriam o mesmo número de assinaturas, sendo este igual a metade das amostras disponíveis por classe (=10). Foram definidas três configurações para os experimentos de reconhecimento, apresentadas na figura 3.

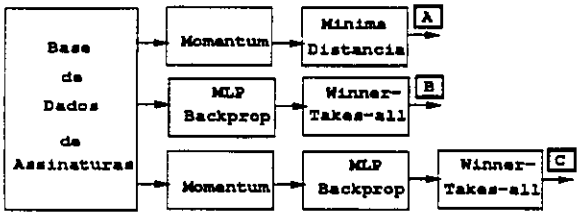


Figura 3: Configurações dos Experimentos

Na configuração [A], Momentum foi utilizado para extração de características, e um classificador de distancia euclidiana mínima no espaço de características foi utilizado para reconhecimento. Foram considerados 6 valores para compor o vetor de características, assim definido: $x = (M_{02}, M_{20}, M_{12}, M_{21}, M_{30}, M_{03})$. Cada classe foi representada pelo vetor de características médio (protótipo) do conjunto de treinamento (equação 6).

$$m_j = \frac{1}{10} \sum_{x \in L_j} x \quad j = 0, 1, \ldots, 6 \qquad (6)$$

Utilizou-se um critério de rejeição com base nos diametros dos conjuntos de treinamento de cada classe. O diametro do conjunto de treinamento é definido como sendo a maior de todas as distancias encontrada entre os padrões do conjunto. Quando a menor distancia $Dmin_j(x_i)$ computada entre um padrão de teste x_i e um dado protótipo m_j for superior a um limiar θ multiplicado pelo diametro δ_j da classe candidata C_j, então deve-se rejeitar o padrão de teste (equação 7). Os testes de reconhecimento com esta configuração foram realizados variando linearmente θ entre 0 e 2, com incremento de 0.01.

$$\text{Rejeitar } x_i \text{ se : } Dmin_j(x_i) > \theta \times \delta_j \qquad (7)$$

Na configuração [B], utilizou-se uma rede neural MLP-Backpropagation de duas camadas escondidas, recebendo diretamente como entrada uma retina de 350 × 90 pontos contendo a imagem de uma assinatura (figura 4). A conectividade entre os neurônios de duas camadas consecutivas é total. A camada de saída foi dimensionada com 7 neurônios (numerados de 0 a 6, cada qual representando uma classe) com o intuito de permitir uma classificação baseada no critério Winner-Takes-All (WTA). Desta forma, na fase de teste, o número da classe reconhecida pela rede é precisamente o número do neuronio que produziu o maior valor na camada de saída. Um critério para rejeição foi definido da seguinte forma: quando a diferença entre as duas maiores saídas $Omax'$, $Omax''$ produzidas pela rede for inferior a um limiar o de rejeição, então deve-se rejeitar o padrão x_i de teste (equação 8). Para a avaliação das taxas de reconhecimento, foram utilizados valores de o variando linearmente entre 0 e 1, com incremento de 0.01. Diferentes valores para o erro de treinamento [3] da rede foram testados. Verificou-se que o valor 0.02 foi o que produziu os melhores resultados.

$$\text{Rejeitar } x_i \text{ se : } |\, Omax' - Omax'' \,| < o \qquad (8)$$

Na configuração [C], acrescentou-se antes da rede neural uma etapa de extração de características, utilizando os mesmos valores de momentum da configuração [A]. Espera-se, com esta configuração, obter-se um ganho em relação ás anteriores, uma vez que combina os resultados das duas

[3] erro de treinamento é maior diferença encontrada entre uma saída desejada e uma saída produzida pela rede, para todos os padrões de treinamento

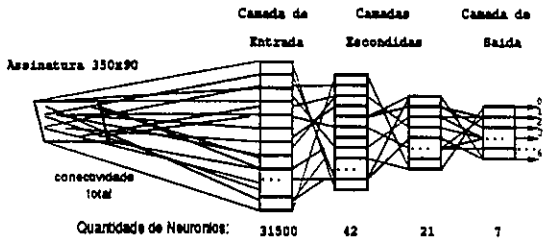


Figura 4: Arquitetura MLP-Backpropagation

Config.	Acerto	Erro	Rejeição	Limiar
A	87.1	0.0	12.9	$\theta = 1.03$
B	82.9	0.0	17.1	$\phi = 0.82$
C	92.9	0.0	7.1	$\varphi = 0.70$

Tabela 1: Índices de acerto (em %) com erro mínimo

técnicas. A arquitetura de rede utilizada foi quase idêntica àquela da figura 4, a memos da camada de entrada, que desta vez possui 6 neurônios (um para cada valor de momentum do vetor de características). O classificador e as estratégias de treinamento e teste foram praticamente os mesmos da configuração anterior. A única diferença é que o erro de treinamento mais adequado foi de 0.1.

6 . Implementação

Todo o código fonte para definição e acesso à base de dados bem como para a implementação de Momentum e dos classificadores foi escrito na linguagem C-Unix. A rede MLP-Backpropagation foi implementada ustilizando o simulador Aspirin/MIGRAINES [6]. Os experimentos foram executados numa estação de trabalho *Sparc 2* da Sun microsystems.

7 Resultados

Foram computados sobre os conjuntos de padrões de teste, os índices médios de acerto, erro e rejeição obtidos pelas diferentes configurações e variação na faixa de valores assumida pelos limiares de rejeição. Na tabela 1, apresentam-se os limiares de rejeição de cada configuração que produziram os menores índices médios de erro. E, na tabela 2, apresentam-se os limiares de rejeição associados aos maiores índices médios de acerto. Na figura 5 são apresentados gráficos evolutivos dos índices de reconhecimento com respeito aos diferentes limiares de rejeição.

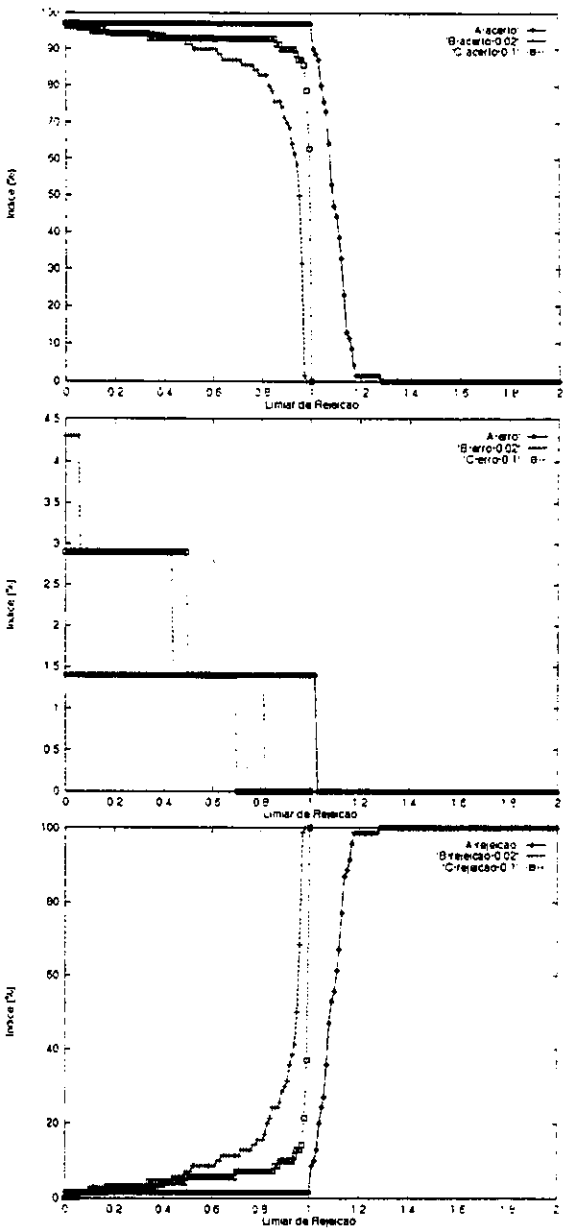


Figura 5: Gráficos evolutivos dos índices de acerto, erro e rejeição

Config.	Acerto	Erro	Rejeição	Limiar
A	97.1	1.4	1.4	$\theta = 1.00$
B	92.9	1.4	5.7	$o = 0.48$
C	97.1	2.9	0.0	$o = 0.05$

Tabela 2: Maiores índices de acerto(em %)

8 Conclusões

Analisando as tabelas 1 e 2, verifica-se que a técnica MLP-BP (configuração [B]) sozinha não produziu bons resultados. Como era esperado, verificou-se que a configuração [C], combinando MLP-BP e Momentum foi a que produziu o melhor índice médio de acerto (92.9%), quando fixando um erro mínimo (tabela 1), e um índice de acerto igual ao da configuração [A] (97.1%), quando o critério de erro mínimo não foi considerado (tabela 2).

Com base nestes resultados, pode-se dizer que a configuração [C] foi mais adequada ao processo de reconhecimento de assinaturas que as demais configurações. Assim, num sistema para reconhecimento off-line de assinaturas, em face da grande variabilidade e complexidade dos padrões, é interessante utilizar um processo de extração de características para reduzir a dimensionalidade dos dados e extrair a informação de interesse, mesmo quando se está utilizando uma técnica supostamente robusta no sentido de abstrair características relevantes dos dados de treinamento e generalizar para os demais padrões, como é o caso de Redes Neurais. Outra conclusão que pode ser extrapolada a partir dos experimentos e resultados é que, considerando os problemas com o processamento de imagens de manuscritos devido a redução drástica das características dinâmicas do sinal, é melhor integrar mais de uma técnica, como na configuração [C], a fim de se obter melhores resultados.

Apesar do estágio inicial da base de dados de assinaturas, as configurações desenvolvidas e os resultados obtidos trazem uma importante contribuição tando para a área de Redes Neurais quanto para a área de reconhecimento de padrões, na medida em que confirmaram experimentalmente as excelentes propriedades de Redes Neurais (MLP-BP), Momentum, e sua combinação no reconhecimento off-line de assinaturas. Espera-se que a construção de uma base de dados mais completa e o desenvolvimento de experimentos para a verificação de assinaturas falsificadas tornem operacionalmente mais práticas contribuições deste trabalho.

Referências

[1] D. L. Bisset, E. C. D. B. C. Filho, and M. C. Fairhurst. A comparative study of neural network structures for practical application in a pattern recognition enviroment. In *Proc. First IEE International Conference on Artificial Neural Networks*, pages 378–382, London, UK, October 1989. IEE.

[2] E. C. de Barros Carvalho Filho. Reconhecimento de assinaturas usando redes neurais artificiais. Master's thesis, Universidade Federal de Pernambuco, 1987.

[3] A. C. P. de Leon Ferreira de Carvalho. Reconhecimento de Seqüências utilizando neuronios booleanos. Master's thesis, Universidade Federal de Pernambuco, 1990.

[4] H. M. Gomes and E. C. D. B. C. Filho. Reconhecimento off-line de assinaturas utilizando momentum. a ser apresentado na XX Conferência Latinoamericana de Informática - CLEIP'94, Cidade do México, México, 1994.

[5] R. Hecht-Nielsen. *Neurocomputing*. Addison-Wesley Publishing Company, 1990.

[6] R. R. Leighton. The Aspirin/MIGRAINES Neural Network Software - Release V6.0. Technical Report MP-91W00050, MITRE Corporation, october 1992. User's Manual.

[7] T. B. Ludemir. Discriminação de Seqüências com redes neuronais digitais. Master's thesis, Universidade Federal de Pernambuco, 1987.

[8] R. Plamondon and G. Lorette. Automatic signature verification and writer identification - the state of the art. *Pattern Recognition*, 5(21):525–537, 1988.

[9] Projeto assinatura eletrônica, maio 1993. Convênio de Cooperação Técnico-Científica firmado entre o Banco do Brasil (CEDIP/Recife) e o Departamento de Informática da UFPE.

[10] R. Prokop and A. Reeves. A survey of moment-based techniques for unocluded object representation and recognition. *CVGIP: Graphical Modules and Image Processing*, 54(5):438–460, september 1992.

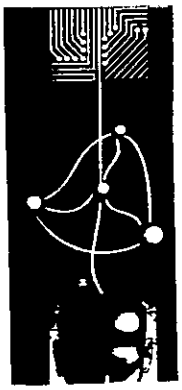
[11] D. E. Rumelhart and R. J. Williams. Learning internal representations by error propagation. *Parallel Distrib. Processing*, 1:318–62, 1986.

[12] A. W. Senior. Off-line handwriting recognition: A review and experiments. Technical Report CUED/F-INFENG/TR 105, Cambridge University Engineering Department, december 1992.

[13] P. K. Simpson. *Artificial Neural Systems*. Pergamon Press, 1990.

[14] G. C. Vasconcelos. Modelo probabilístico da dinâmica de recuperação de erros em redes neurais booleanas. Master's thesis, Universidade Federal de Pernambuco, 1991.

[15] P. D. Wasserman. *Neural Computing: Theory and Pratice*. Van Nostrand Reinhold, 1989.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajuba. 24 a 27 de outubro de 1994

Reconhecimento de Caracteres Manuscritos Utilizando Redes Neurais

*Edna L. Flôres, *Eder N. Rezende, *Gilberto A. Carrijo e **João B. T. Yabuti.

*Departamento de Engenharia Elétrica
Universidade Federal de Uberlândia
38.400-902 - Uberlândia - MG .

**Departamento de Comunicações
Universidade Estadual de Campinas
13.081 - 970 - Campinas - SP.

Resumo - Este artigo trata do reconhecimento de caracteres escrito à mão através de redes neurais. Para chegarmos a um resultado satisfatório os caracteres são normalizados, tornando-se insensíveis à escala, translação e rotação. O artigo consta de 4 tópicos principais, onde no primeiro trata-se das técnicas de reconhecimento; no segundo é exposto a técnica utilizada para tornar os caracteres invariantes à escala, translação e rotação; no terceiro tópico trata-se brevemente dos algoritmos utilizados para o reconhecimento e descreve-se os resultados obtidos. No quarto tópico são feitas conclusões a respeito dos resultados.

Palavras Chaves - Caracteres Manuscritos, Escala, Translação, Rotação, Reconhecimento.

1 - INTRODUÇÃO

Vários centros de pesquisas e Universiadades estão tentando desenvolver técnicas de reconhecimento de caracteres manuscritos que tenham um nível de desempenho práticamente igual ao do homem. O motivo disto é a grande importância nas aplicações científicas e econômica, tais como: reconhecimento de código de endereçamento postal, leitura automática de cheques bancários, sistemas de tarifação telefônica, sistemas de leitura automática para cegos, etc. A obtenção de um software que consiga fazer o papel do homem pode representar uma grande econômia de tempo e um bom ganho para as empresas. O reconhecimento de caracteres possue 3 passos principais: aquisição da imagem, segmentação, e finalmente classificação dos caracteres. No processo de aquisição da imagem os caracteres são obtidos através de um processo óptico qualquer, tal como em "scanner". A Segmentação dos caracteres é o processo no qual os caracteres que aparecem conectados são separados. Classificação é o processo pelo qual os caracteres são comparados a padrões já existentes para o seu reconhecimento.

2-ESCALA,TRANSLAÇÃO E ROTAÇÃO [1]

O reconhecimento de caracteres manuscritos utilizando redes neurais não apresenta resultados satisfatórios, se usarmos caracteres no treinamento diferentes daqueles usados no reconhecimento. Ocorre o mesmo quanto a translação, se a rede for treinada com caracteres em uma posição e no reconhecimento estes estiverem situados em outras posições. Quanto a rotação, isso também ocorre, se os caracteres são treinados inclinados com um determinado ângulo e no reconhecimento estes estiverem com outros ângulos de inclinação. Em todos os casos o reconhecimento fica bastante prejudicado. O algorítmo [1] desenvolvido para tornar o caracter invariante à escala, translação e rotação segue os seguintes passos:

INVARIÂNCIA A ESCALA

O algorítmo permite que o caracter seja ampliado ou reduzido.

Define-se:

n_l - número de linhas da imagem.
n_c - número de colunas da imagem.
n_{ln} - número de linhas da imagem na nova escala.
n_{cn} - número de colunas da imagem na nova escala.
e_l - fator de escala das linhas.
e_c - fator de escala das colunas.
e_f - fator de escala final da imagem.

$$e_l = \frac{n_{ln}}{n_l}, \quad e_c = \frac{n_{cn}}{n_c} \qquad (1)$$

O fator de escala final que a imagem fica submetida é calculado do seguinte modo:

. Se a diferença entre e_l e e_c for menor que 0.4, o e_f será igual a média entre e_l e e_c.

. Se e_l for maior ou igual a 1 e e_c menor ou igual a 1, o e_f será o maior dos dois termos.

. Se e_l e e_c forem maior ou igual a 1, o e_f será o menor dos dois temos.

. Se e_l for maior ou igual a 2 o e_f será igual a e_c, caso contrário será e_l.

A imagem invariante a escala será:

$$f_g(x_i, y_j) = f(e_f \cdot x_i, e_f \cdot y_j) \qquad (2)$$

onde $f(x_i, y_j)$ fornece o valor do "pixel" da coordenada (x, y).

INVARIÂNCIA A TRANSLAÇÃO

A translação é realizada utilizando a matriz resultante do processo de escala.
Para que o caracter fique insensível a translação é calculado seu centro de gravidade e deslocado este para a origem do novo sistema de coordenadas que é o centro da matriz que contém o caracter.
O centro de gravidade é calculado pela média das coordenadas x e y dos "pixels-on", de acordo com as fórmulas abaixo:

$$P = \sum_{i=1}^{N} \sum_{j=1}^{N} f_g(x_i, Y_j) \qquad (3)$$

onde P é o número de "pixels-on" da matriz do caracter.

O centro de gravidade será:

$$x_{cg} = \frac{1}{P} \sum_{i=1}^{N} \sum_{j=1}^{N} f_g(x_i, y_j) . x_i \qquad (4.a)$$

$$y_{cg} = \frac{1}{P} \sum_{i=1}^{N} \sum_{j=1}^{N} f_g(x_i, y_j) . y_j \qquad (4.b)$$

O deslocamento do centro de gravidade do caracter para a origem do novo sistema de coordenadas (centro de gravidade da matriz) é calculado da seguinte maneira:

$$x_{cgm} = x_{cg} - x_o, \quad y_{cgm} = y_{cg} - y_o \qquad (5)$$

onde:

x_o - abcissa do centro da matriz do caracter.
y_o - ordenada do centro da matriz do caracter.
x_{cgm} - abcissa do centro de gravidade da matriz.
y_{cgm} - ordenada do centro de gravidade da matriz.

A imagem invariante a translação será:

$$f_{ET}(x_i, y_j) = f_E(x_i + x_{cgm}, y_j + y_{cgm}) \qquad (6)$$

INVARIÂNCIA A ROTAÇÃO

A rotação é realizada utilizando a matriz resultante do processo de translação.
O caracter fica invariante à rotação rodando-o de maneira que a direção da variância máxima coincide com o eixo x. A derivaçao da função é baseada na transformação de Karhunen-Loéve que tem sido usada em algumas aplicações [2, 3, 4]. Esta transformação explica o seguinte: dado um conjunto de vetores, o autovetor que corresponde ao maior autovalor da matriz de covariância calculada do conjunto de vetores, pontos na direção da variância máxima [3,4]. Esta propriedade pode ser usada para manter invariância a rotação desde que a deteção da direção da variância máxima revelará o ângulo de rotação. A solução geral para qualquer tamanho de vetores será impraticável. Para vetores de duas dimensões formados pelas coordenadas x e y de "pixels-on", os autovalores são fáceis para calcular e uma fórmula para o autovetor correspondendo ao maior autovalor pode ser derivada da matriz de covariância 2 x 2. Os parâmetros da rotação são derivadps como segue:

Define-se:

$$\begin{bmatrix} m_x \\ m_y \end{bmatrix} = \frac{1}{\sum_{i=1}^{N}\sum_{j=1}^{N} f_{ET}(x_i, y_j)} \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ET}(x_i, y_j) \begin{bmatrix} x_i \\ y_j \end{bmatrix} \qquad (7)$$

$$P = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ET}(x_i, y_j) \qquad (8)$$

$$T_{xx} = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ET}(x_i, y_j) \cdot x_i^2 \qquad (9)$$

$$T_{yy} = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ET}(x_i, y_j) \cdot y_j^2 \qquad (10)$$

$$T_{xy} = \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ET}(x_i, y_j) \cdot x_i \cdot y_j \qquad (11)$$

A matriz de covariância é definida como:

$$C = \frac{1}{P}\left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} m_x \\ m_y \end{bmatrix}\right)\left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} m_x \\ m_y \end{bmatrix}\right)^T \qquad (12)$$

Pode ser simplificada para:

$$C = \left(\frac{1}{\sum_{i=1}^{N}\sum_{j=1}^{N} f_{ET}(x_i, y_j)} \sum_{i=1}^{N}\sum_{j=1}^{N} f_{ET}(x_i, y_j) \begin{bmatrix} x_i \\ y_j \end{bmatrix}\begin{bmatrix} x_i \\ y_j \end{bmatrix}^T\right) + -\left(\begin{bmatrix} m_x \\ m_y \end{bmatrix}\begin{bmatrix} m_x \\ m_y \end{bmatrix}^T\right) \qquad (13)$$

Desde que a invariância a translação têm sido mantida, as médias m_x e m_y são zero. Além disso, o termo médio na frente da matriz pode ser eliminado desde que ele não muda a direção dos autovetores. Portanto, a matriz de covariância torna-se:

$$C = \begin{bmatrix} T_{xx} & T_{xy} \\ T_{xy} & T_{yy} \end{bmatrix} \quad (14)$$

Finalmente, o seno e cosseno do ângulo de rotação são:

$$\sin\theta = \frac{(T_{yy} - T_{xx}) + \sqrt{(T_{yy} - T_{xx})^2 + 4.T_{xy}^2}}{\sqrt{(8.T_{xy}^2 + D\sqrt{((T_{yy} - T_{xx})^2 + 4.T_{xy}^2)}}} \quad (15)$$

$$\cos\theta = \frac{2.T_{xy}}{\sqrt{(8.T_{xy}^2 + D\sqrt{((T_{yy} - T_{xx})^2 + 4.T_{xy}^2)}}} \quad (16)$$

onde :

$$D = 2.(T_{yy} - T_{xx})^2 + 2.(T_{yy} - T_{xx}) \quad (17)$$

A imagem invariante à rotação será:

$$f_{ETR}(x_i, y_j) = f_{ET}(\cos\theta.\, x_i - \sin\theta.\, y_j,\ \sin\theta.\, x_i + \cos\theta.\, y_j) \quad (18)$$

3 - Redes Neurais

3.1 - Algoritmo Back-Propagation

O algoritmo utilizado para o reconhecimento de caracteres manuscritos foi o back-propagation. Este algoritmo [5] é um processo de treinamento iterativo no qual o sinal erro de saída é propagado posterior através da rede e é usado para modificar os valores dos pesos. A Figura 1 mostra a rede neural de 3 camadas, utilizada para o reconhecimento de caracteres manuscritos.

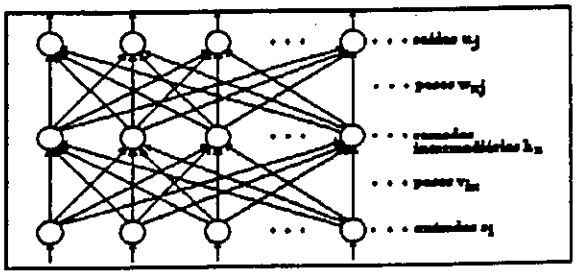


Fig. 1 - Rede neural de 3 camadas.

3.2 - Resultados

Os "pixels" de cada caracter treinado e reconhecido pela rede, invariantes à escala, translação e rotação, são colocados em uma matriz 30x30, logo o número de entradas da rede para cada caracter é 900. O treinamento é realizado para os números de 0 a 9, constituindo assim um arquivo de treinamento, portanto o número de saídas da rede é 10. O número de camadas intermediárias utilizadas para treinamento é 20. A figura 2 mostra os 10 caracteres numéricos de uma pessoa utilizados antes de ficarem invariantes à escala, translação e rotação; a figura 3 mostra os 10 caracteres numéricos de uma pessoa utilizados para treinamento e reconhecimento que são invariantes à escala, translação e rotação.

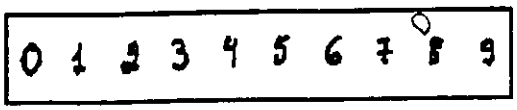


Fig. 2 - Caracteres numéricos de uma pessoa antes de ficarem invariantes à escala, translação e rotação.

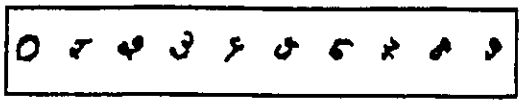


Fig. 3 - Caracteres numéricos de uma pessoa depois de ficarem invariantes à escala, translação e rotação.

Foram feitos 3 tipos de testes com caracteres de dimensões 30x30. O primeiro foi feito com os caracteres apenas de uma pessoa e utilizando para treinamento 10 arquivos cada um contendo 10 números decimais para treinamento. Trinta outros arquivos da mesma pessoa e diferentes daqueles usados para o treinamento foram utilizados no reconhecimento. A tabela 1 e a figura 4 apresenta os resultados obtidos.

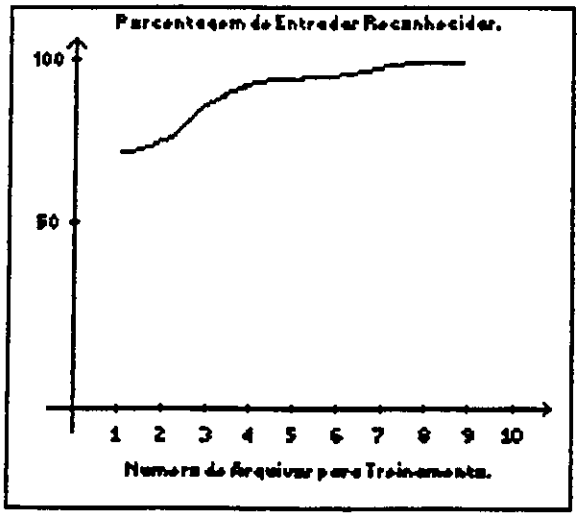


Fig. 4 - Gráfico da porcentagem de entradas reconhecidas x número de arquivos de uma pessoa para treinamento.

Verifica-se que quanto maior o número de arquivos usados no treinamento a porcentagem aumenta.

O segundo teste foi feito com 4 pessoas diferentes. Inicialmente foi utilizado um arquivo contendo 10 números decimais de cada pessoa para o treinamento. No reconhecimento foi utilizado outros 10 arquivos das mesmas pessoas, totalizando 40 arquivos ou um total de 400 caracteres. Neste caso obtivemos um reconhecimento de 56.25%. O mesmo teste foi feito tomando-se 2, 3 e 4 arquivos de cada uma das 4 pessoas acima mencionadas. A porcentagem de acerto aumentou gradativamente atingindo 71.00% de acerto para o caso de 4 arquivos de cada pessoa. Em todos os testes foram utilizados no reconhecimento 10 arquivos de cada pessoa. Os resultados são mostrados na tabela 2 e figura 5.

Tabela 1 - Treinamento e reconhecimento com arquivos de uma pessoa.

Número de Arquivos para Treinamento	Número de Entradas Reconhecidas	Porcentagem de Entradas Reconhecidas
1	208	69.30%
2	219	73.00%
3	265	83.30%
4	266	88.87%
5	271	90.33%
6	272	90.67%
7	280	93.33%
8	284	94.67%
9	283	94.33%
10	285	95.00%

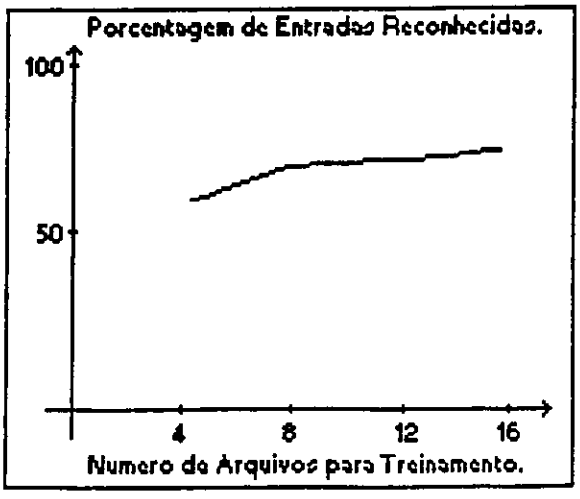


Fig. 5 - Gráfico da porcentagem de entradas reconhecidas x número de arquivos de 4 pessoas para treinamento.

O terceiro teste foi feito treinando-se a rede com 4 arquivos contendo cada um 10

números decimais de 4 pessoas distintas. Para o reconhecimento foram utilizados um total de 40 arquivos de pessoas diferentes daquelas usadas no treinamento, tentando fazer um sistema reconhecedor independente da pessoa. Neste caso a porcentagem de acerto caiu bastante, o seu valor foi de 57.75%. Em todos os testes anteriores usamos caracteres tendo uma dimensão de 30 x 30.

Quando usamos caracteres de dimensões 20 x 20 percebemos uma diminuição na porcentagem de acerto, como por exemplo a porcentagem de acerto foi de 88% quando usamos 5 arquivos de uma única pessoa para treinamento . Este valor foi de 90.33% quando se usou caracteres de dimensões 30 x 30.

Quando aumentamos a dimensões dos caracteres para 40 x 40, e fazendo-se o mesmo teste anterior a porcentagem de acerto passou para 93.00%.

Tabela 2 - Treinamento e reconhecimento com arquivos de 4 pessoas distintas das usadas no treinamto.

Número de Arquivos para Treinamento	Número de Entradas Reconhecidas	Porcentagem de Entradas Reconhecidas
4	225	56.25%
8	267	66.75%
12	276	69.00%
16	284	71.00%

Conclusões:

A rede funcionou satisfatóriamente quando foi utilizado os arquivos de uma única pessoa para treinamento e reconhecimento. Como era de se esperar quanto maior o número de arquivos utilizados para treinamento melhores foram os resultados. O mesmo ocorre quando se utiliza 4 pessoas distintas para treinamento. Podemos verificar que a porcentagem de acerto cai um pouco em relação a citada anteriormente, e concluímos que quanto maior o número de arquivos utilizados no treinamento, maior a porcentagem de reconhecimento. Quando foram usados 40 arquivos de pessoas diferentes daquelas usadas no treinamento, a porcentagem cai bastante, mas alguns testes estão sendo feitos para melhorar os resultados. Na utilização de caracteres de dimensões 20 x 20 diminui-se a porcentagem de reconhecimento em relação a dos caracteres de dimensões 30 x 30 como era de se esperar, isso devido a menor quantidade de "pixels" utilizados no treinamento. Quando foi aumentada as dimensões dos caracteres para 40 x 40 podemos verificar que a porcentagem de acerto aumentou como era de se esperar, isto porque no treinamento foram utilizados mais "pixels" nas matrizes dos caracteres.

Referências

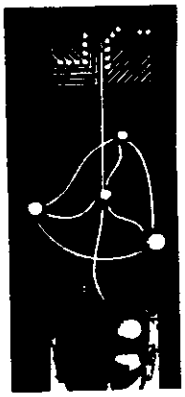
1 - C. Yüceer and K. Oflazer, A Rotation, Scaling, and Translation Invariant Pattern Classification System, Pattern Recognition 26 (5), pp. 687 - 710 (May 1993).

2 - H. A. Malki and A.Moghaddamjoo, Using the Karhunen-Loéve transformations in the back-propagation training algorithm, IEEE Trans. Neural Networks 2 (1), pp. 162 - 165 (January 1991).

3 - M. Kirby and L. Sirovich, Application of the Karhunen-Loéve procedures for the characterization of human faces, IEEE Trans. Pattern Analysis Mach. Intell. 12 (1), pp. 103 - 108 (January 1990).

4 - R. C. Gonzales and P. Wintz, Digital Image Processing, pp. 122 - 130. Addison - Wesley, Reading, Massachusetts (1987).

5 - D. J. Burr, Experiments on Neural Net Recognition of Spoken and Written Text, IEEE Trans. on Acoustics, Speech, and Signal Processing 36 (7), 1162 - 1168 (July 1988).



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá. 24 a 27 de outubro de 1994

PROCESSAMENTO NEURAL-ADAPTATIVO DE SINAIS

João Batista Destro Filho e João Marcos Travassos Romano

Departamento de Comunicações/FEE - UNICAMP - Caixa Postal 6101
Cidade Universitária - Campinas - SP - 13081-970
e-mail: destro@decom.fee.unicamp.br

RESUMO - Este trabalho propõe uma abordagem unificada das técnicas de redes neurais e da filtragem adaptativa, com o objetivo de desenvolvê-las simultaneamente. Tal abordagem está baseada na inter-relação existente entre as redes neurais e a filtragem adaptativa, a qual pode ser evidenciada por uma série de analogias matemáticas e conceituais entre os dois campos. Apresentam-se alguns resultados preliminares já alcançados, que consistem na análise matemática do processamento paralelo distribuído associado a uma rede Perceptron multi-camadas e na proposição de um algoritmo alternativo para a adaptação da cascata de filtros adaptativos transversais, cujo desempenho é avaliado experimentalmente para um caso simples.

1. INTRODUÇÃO.

As redes neurais correspondem, atualmente, a uma das técnicas mais promissoras em diversas áreas (por exemplo, na robótica [1]), graças às suas propriedades coletivas emergentes. Por outro lado, segundo Amari [1], "a matemática das redes neurais encontra-se ainda em sua infância". De fato, constata-se atualmente uma certa dificuldade de compreensão matemática do processamento paralelo distribuído [2] e da auto-organização [3], princípios básicos do processamento de informação neural. Isto explica, de certa forma, o caráter empírico do projeto de redes neurais, bem como seu aprendizado complexo e lento.

As técnicas de processamento de sinais (e, dentre estas, a filtragem adaptativa), já são aplicadas industrialmente em diversas áreas, por exemplo, nas telecomunicações e na geofísica [4]. Isto pode ser justificado pelo seu sólido embasamento matemático, que compreende, dentre várias teorias, o controle automático e a teoria estatística do sinal. Entretanto, a filtragem adaptativa também possui algumas limitações, por exemplo, baixo desempenho em aplicações que envolvam ruído não-gaussiano e não-aditivo.

Conclui-se, portanto, que as redes neurais e a filtragem adaptativa correspondem a técnicas com vantagens e fraquezas complementares. De um lado, as propriedades coletivas emergentes neurais, de difícil análise matemática; de outro, os filtros adaptativos com seu sólido formalismo matemático e limitações práticas.

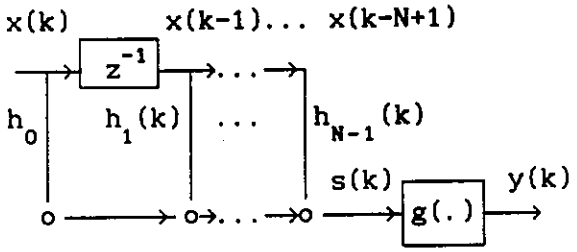
O objetivo deste trabalho é propiciar elementos para a pesquisa unificada das duas áreas, bem como evidenciar como esta pode ser útil para ambas. Tal pesquisa é aqui denominada de "Processamento Neural-Adaptativo de Sinais". Corresponde

a analisar simultaneamente, de forma teórica e experimental, estruturas matematicamente equivalentes de redes neurais e de filtros adaptativos. Desta forma, objetiva-se alcançar contribuições para ambas as áreas, que representem resultados alternativos àqueles da pesquisa tradicional.

Na seção 2, apresentam-se várias analogias conceituais e matemáticas, que evidenciam a existência de uma inter-relação entre as redes neurais e a filtragem adaptativa. Na seção 3, analisa-se simultaneamente um Perceptron multi-camadas linear, parcialmente interconectado, e a cascata de filtros adaptativos, propondo-se uma expressão matemática para o processamento paralelo distribuído desta rede e um algoritmo alternativo para o treinamento da cascata. Tais hipóteses são verificadas experimentalmente na seção 4. As principais conclusões deste trabalho são apresentadas na seção 5.

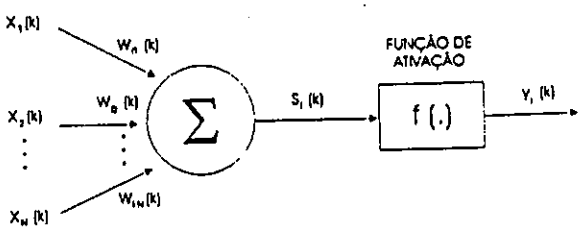
2. A INTER-RELAÇÃO EXISTENTE ENTRE AS REDES NEURAIS E A FILTRAGEM ADAPTATIVA.

ANALOGIA 1: Filtro não-linear adaptativo e neurônio Perceptron.



$$y(k) = g(s(k)) = g\left(\sum_{j=0}^{N-1} h_j(k).x(k-j) \right) \qquad (1)$$

Figura 1: Filtro não-linear adaptativo - ordem N.



$$y_i(k) = f(s_i(k)) = f\left(\sum_{j=1}^{N} w_{ij}(k).x_j(k) \right) \qquad (2)$$

Figura 2: Neurônio Perceptron.

A grandeza g(.) (fig. 1) representa a não-linearidade do filtro adaptativo. A comparação das figs. 1-2 e das eqs. (1) - (2) permite concluir que as duas estruturas são análogas, sendo possível estabelecer uma equivalência entre g(.) e a função de ativação f(.), bem como entre os coeficientes h_j do filtro e os pesos sinápticos w_{ij} do neurônio.

ANALOGIA 2: Equalizador adaptativo de Bussgang e neurônio Perceptron treinado de forma não-supervisionada.

Esta analogia representa um caso especial da analogia 1. O equalizador de Bussgang [4] corresponde a um filtro não-linear adaptativo, idêntico àquele mostrado pela fig. 1. Sua relação de entrada-saída é exatamente a mesma da eq. (1), sendo que a não-linearidade g(.) será aqui denominada de "estimador de Bussgang".

O propósito de aplicação desta estrutura está mostrado na fig. 3.

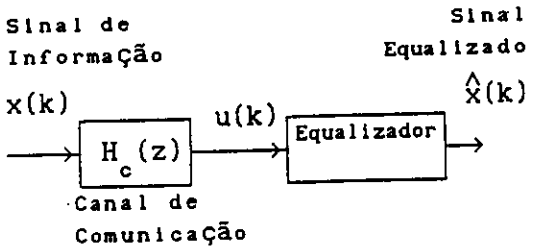


Fig. 3: Operação de um equalizador.

O objetivo é inverter a função de transferência do canal de comunicação (representada por $H_c(z)$), para gerar a melhor estimativa possível do sinal de informação x(k) transmitido. Isto é realizado de forma auto-organizativa. Portanto, através da analogia 1, conclui-se que o equalizador de Bussgang corresponde a um neurônio Perceptron treinado de forma não-supervisionada.

Deve-se notar que o estimador de Bussgang pode ser deduzido matematicamente [4] e depende da relação sinal-ruído do sistema de comunicação. Se este parâmetro possuir amplitude reduzida, o estimador de Bussgang assume a forma de uma sigmóide, exatamente como a função de ativação de um neurônio Perceptron.

ANALOGIA 3: Filtro de erro de predição linear treinado pelo algoritmo LMS e neurônio de Kohonen [3].

Estas estruturas estão respectivamente mostradas nas figs. 4 e 5.

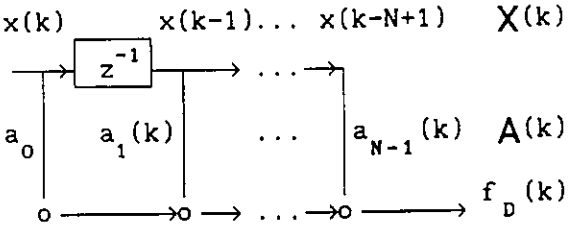


Figura 4: Filtro de erro de predição linear - ordem N.

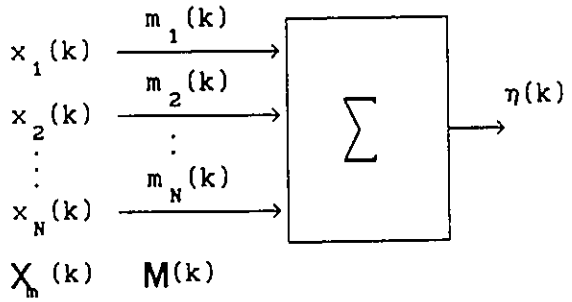


Fig. 5: Neurônio de Kohonen [3].

Apresentam-se a seguir as equações de filtragem e de adaptação de cada estrutura.

Filtro de erro de predição:

$$f_D(k) = \sum_{j=0}^{N-1} a_j(k).x(k-j) \quad (3)$$

$$A(k+1) = A(k) - \mu.f(k).X(k) \quad (4)$$

$$X(k)^T = [x(k)\ x(k-1)\ \ldots\ x(k-N+1)] \quad (5)$$

$f_D(k)$: erro de predição linear.
$A(k)$: vetor de coeficientes.
μ : passo de adaptação.
$X(k)$: vetor de dados de entrada.
A eq.(4) corresponde ao algoritmo do gradiente estocástico (ou LMS).

Neurônio de Kohonen:

$$\eta(k) = \sum_{j=1}^{N} m_j(k).x_j(k) \quad (6)$$

$$M(k+1) = M(k) - \alpha.\eta(k).X_m(k) \quad (7)$$

$M(k)$: vetor de pesos sinápticos.
α : constante real positiva.

As equações de filtragem (3) e (6), bem como as equações de treinamento (4) e (7), são claramente idênticas. Além disso, demonstra-se em [5] a igualdade das condições de convergência dos algoritmos expressos pelas eqs. (4) e (7).

ANALOGIA 4: Cascata de filtros adaptativos transversais e Perceptron multi-camadas (ou MLP).

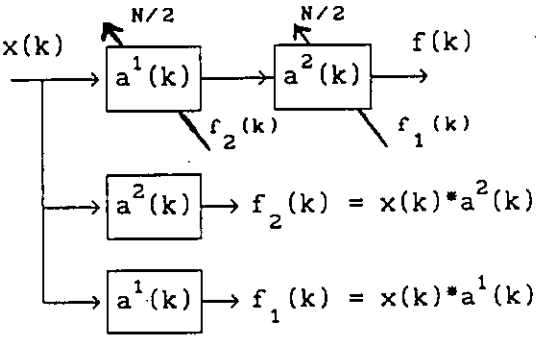


Fig. 6: Filtro de erro de predição linear, ordem N - FORMA CASCATA.

A fig. 6 apresenta dois filtros de erro de predição linear, cada

qual de ordem N/2, conectados em série. Tal estrutura, denominada "realização cascata", desempenha a mesma função que o filtro de erro de predição linear da fig. 4, de ordem N, denominado de "forma direta".

O símbolo "*" representa a operação de convolução discreta, enenquanto $a^1(k)$ é a resposta ao impulso do filtro 1. Os termos $f_1(k)$ e $f_2(k)$ são denominados "sinais de gradiente", pois são utilizados no treinamento da cascata através do algoritmo do gradiente estocástico na forma cascata (ou LMSc) [6].

Algoritmo LMSc

$$A_1(k+1) = A_1(k) - \mu . f(k) . F_2(k) \quad (8a)$$

$$A_2(k+1) = A_2(k) - \mu . f(k) . F_1(k) \quad (8b)$$

$A_i(k)$: vetor de coeficientes do filtro i ($i = 1,2$).

μ : passo de adaptação.

$$F_i(k)^T = [f_i(k-1)\ f_i(k-2)\ \ldots\ f_i(k-(N/2))];\ i = 1,2 \quad (9)$$

$$f_i(k) = A_i(k)^T . X_p(k);\ i = 1,2 \quad (10)$$

$$X_p(k)^T = [x(k)\ x(k-1)\ \ldots\ x(k-(N/2))] \quad (11)$$

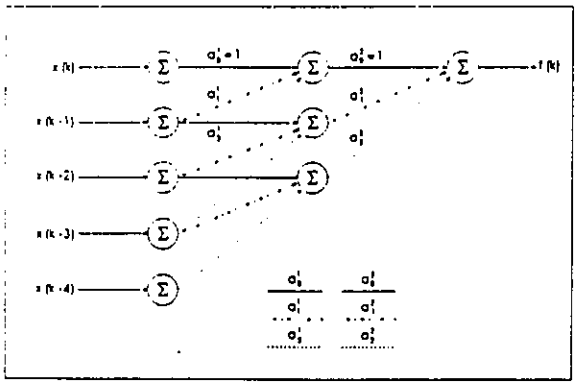


Fig.7: Perceptron multi-camadas linear e parcialmente conectado [6].

A rede neural da fig. 7, proposta em [6], possui função de ativação linear, é parcialmente interconectada e seus vetores de treinamento $X(k)$ possuem componentes que representam amostras adjacentes de uma série temporal x(k):

$$X(k)^T = [x(k)\ x(k-1)\ \ldots\ x(k-N)] \quad (12)$$

Obs.: N = 4 na figura 7.

Pode-se demonstrar [6] que as equações de filtragem e do algoritmo LMSc (eqs.(8a-b)), para a cascata da fig. 6, são respectivamente iguais às equações de filtragem e do algoritmo de retropropagação [2], para a rede neural mostrada na fig. 7.

Ou seja, as figs 6 e 7 constituem estruturas matematicamente equivalentes. Em particular, deve-se notar que a primeira e a segunda camadas do Perceptron multi-camadas da fig. 7 podem ser respectivamente associadas aos filtros 1 e 2 da cascata (fig. 6). Isto porque os pesos sinápticos dos neurônios situados em uma mesma camada são iguais aos coeficientes do respectivo filtro associado.

ANALOGIA 5: Sinapse de Hebb e algoritmo LMS para a predição linear.

A fig.8 apresenta em detalhe a sinapse biológica existente entre a terminação j do axônio de um neurônio A e o dendrito de seu vizinho B.

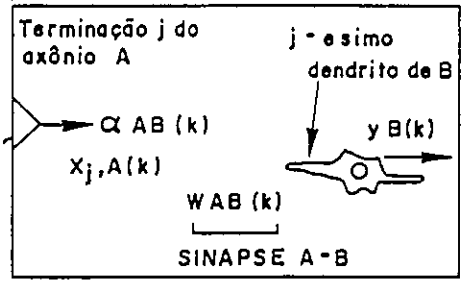


Fig. 8: Sinapse de Hebb.

A lei de Hebb pode ser assim enunciada [7]: *"A sinapse A-B será cada vez mais eficiente à medida que um processo simultâneo de excitação intensa, proveniente de A, gerando atividade neural de saída intensa em B, persistir repetida-repetidamente".*

"Eficiência" significa aqui que, por menor que seja a amplitude da excitação proveniente do neurônio A, esta influencia cada vez mais a atividade neural de saída de B, em

relação às demais excitações externas que atingem o neurônio B.

Definem-se agora algumas grandezas matemáticas (vide fig. 8).

$w_{AB}(k)$: eficiência da sinapse A-B no instante k.

$x_{j,A}(k)$: amplitude da excitação externa, proveniente de A.

$y_B(k)$: amplitude da atividade neural de saída de B.

α_{AB}: : coeficiente de plasticidade da sinapse A-B.

α_{AB} é uma constante real que evidencia o quanto os parâmetros bioquímicos, controladores do mecanismo de sinapse entre os neurônio A e B, são susceptíveis a variações.

Desta forma, a lei de Hebb pode ser expressa matematicamente pela seguinte equação [3,5]:

$$w_{AB}(k+1) = w_{AB}(k) + \alpha_{AB}.x_{j,A}(k).y_B(k) \qquad (13)$$

Reescrevendo o algoritmo do gradiente estocástico, aplicado à predição linear (eq.(4)), considerando-se apenas um único coeficiente:

$$a_j(k+1) = a_j(k) - \mu.x_j(k).f(k) \qquad (14)$$

As equações (13) e (14) são claramente análogas.

3. ANÁLISE MATEMÁTICA DO PROCESSAMENTO PARALELO DISTRIBUÍDO (ou PDP)

Analisam-se nesta seção as estruturas mostradas nas figs. 6 e 7, no contexto da analogia 4.

O processamento paralalelo distribuído (PDP) do Perceptron multi-camadas da fig.7 corresponde à interação entre as duas camadas desta estrutura. Ou seja, de acordo com a analogia 5, o PDP corresponde à interação dos dois filtros da cascata da fig.6. Tal interação, por sua vez, é função dos sinais de gradiente. Isto pode ser concluído a partir da análise das equações do algoritmo LMSc (eqs.(8)). Por exemplo, reescrevendo a eq.(8a), tem-se que:

$$\Delta A_1(k) = A_1(k+1) - A_1(k) \qquad (15)$$

$$\Delta A_1(k) = -\mu.f(k).F_2(k) \qquad (16)$$

De acordo com a eq.(16), a variação nos coeficientes do filtro 1 é diretamente proporcional ao sinal de gradiente 2.

Denotando o PDP da rede neural pelo sinal p(k), uma primeira aproximação para este poderia ser [5]:

$$p(k) \cong f_1(k) + f_2(k) \qquad (17)$$

Desde que a estrutura da fig.6 é análoga ao Perceptron multi-camadas, conclui-se que o PDP constitui elemento intrínseco da cascata de filtros adaptativos. Desta forma, considerando que este sistema é variante no tempo e que realiza PDP, pode-se considerar que a saída da cascata é função não apenas da convolução discreta entre a entrada x(k) e as respostas ao impulso dos filtros 1 e 2, como também de p(k). Neste contexto, uma primeira aproximação para a saída da cascata poderia ser [5]:

$$f_p(k) \cong f(k) + p(k) \qquad (18)$$

Onde $f_p(k)$ denota o sinal de saída da fig.6, sob a ótica de redes neurais, a ser considerado durante a adaptação da cascata.

Definindo-se uma função de custo apropriada com base nas eqs. (17) e (18), é possível deduzir um algoritmo alternativo para o treinamento da cascata. Isto é realizado em [5] e tal algoritmo, denominado gradiente estocástico neural na forma cascata (ou NLMS) é apresentado logo a seguir.

Algoritmo NLMS

$$A_1(k+1) = A_1(k) - \mu.f(k).F_2(k) - \mu.f(k).X_p(k) \qquad (19)$$

A eq.(18) representa um sistema que não obedece ao princípio da superposição [5], justamente devido

ao termo p(k). Desta forma, o PDP pode ser considerado uma espécie de não-linearidade sistêmica, independentemente da função de ativação.

4. RESULTADOS DE SIMULAÇÃO

Com o objetivo de avaliar a validade das aproximações das eqs. (17)-(18), simularam-se o filtro direto da fig.4 (treinado pelo algoritmo LMS - eq.(4) -) e a cascata da fig.6, para N = 4 (treinada pelo algoritmo LMSc - eqs. (8a-b) - e pelo NLMS - eq.(19)-). Tais estruturas foram utilizadas para a predição linear de um processo estocástico auto-regressivo (AR), obtido pela filtragem de um ruído branco gaussiano por uma estrutura recursiva de ordem 4.

A fig. 9 apresenta as curvas de erro quadrático médio para o três algoritmos simulados. Tais curvas correspondem à média para vários modelos AR e para várias condições iniciais impostas ao sistema. A tabela 1 apresenta o desvio padrão s(.) da grandeza IT, definida como sendo a quantidade média de iterações para o algoritmo considerado convergir. s(IT) caracteriza, portanto, a dependência do algoritmo considerado relativamente à inicialização do sistema. Quanto menor s(IT), menor esta dependência.

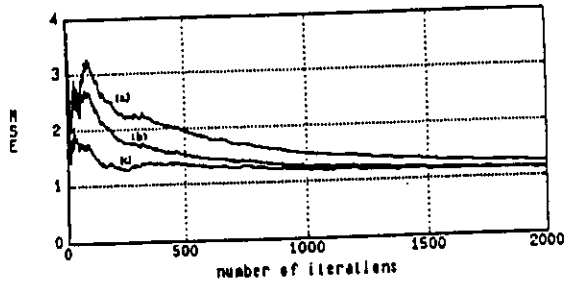


Fig. 9: Dinâmica dos algoritmos. (a) LMSc; (b) NLMS; (c) LMS.

TABELA 1

Algoritmo	LMS	LMSc	NLMS
s(IT)	248	518	439

O algoritmo NLMS, além de convergir mais rapidamente que sua versão original (o LMSc), é menos dependente das condições iniciais, de acordo com a tabela 1.

5. CONCLUSÃO

Evidenciou-se neste trabalho a existência de uma inter-relação entre as redes neurais e a filtragem adaptativa. Através do estudo de uma das analogias apresentadas, foi possível, simultaneamente, analisar o processamento paralelo distribuído de um Perceptron multi-camadas linear e propor um algoritmo alternativo para o treinamento da cascata de filtros adaptativos, o qual se mostrou mais eficiente que sua versão original para a simulação de um caso simples. Estes resultados preliminares evidenciam novos horizontes através da pesquisa conjunta de redes neurais e da filtragem adaptativa.

6. REFERÊNCIAS

[1] S.-I. Amari; "Mathematical Foundations of Neurocomputing", Proc. IEEE, vol.78, no. 9, pp. 1443 - 1463, Sep. 1990.

[2] D.E. Humelhart, J.L. McClelland and the PDP Research Group; "Parallel Distributed Processing: Explorations in the Microstructure of Cognition", vol.1, MIT Press, 1989.

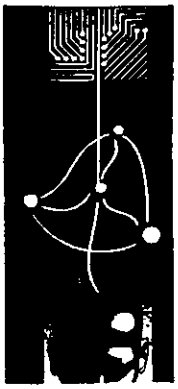
[3] T. Kohonen; "Self-Organization and Associative Memory", Springer-Verlag, Berlin, 1989.

[4] S. Haykin; "Adaptive Filter Theory", Prentice-Hall Inc., New Jersey, 1991.

[5] J.B. Destro Filho; "Base Teórica para o Processamento Neural-Adaptativo de Sinais", tese de de mestrado, FEE-UNICAMP, Junho 1994.

[6] S. Marcos et alli; "A Unified Framework for Gradient Algorithms used for Filter Adaptation and Neural Network Training", Int. J. of Circuit Theory and Applications, vol. 20, pp. 159 - 200, 1992.

[7] D.O. Hebb; "The Organization of Behavior", John Wiley & Sons, New York, 1949.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá. 24 a 27 de outubro de 1994

Uso da Quantização Vetorial e Mapas de Kohonen para Traçado de Trajetórias do Sinal da Fala

Ana Cristina S. Antunes
Gilberto A. Carrijo

Departamento de Engenharia Elétrica
Universidade Federal de Uberlândia
38.400-902 - Uberlândia - MG

Resumo. Desenvolveu-se um trabalho com o objetivo de analisar os melhores métodos para visualização dos sinais da fala, distribuindo-os sobre mapas fonotópicos. Utilizou-se ora um algoritmo desenvolvido por Teuvo Kohonen, o qual baseia-se na organização dos sistemas nervosos biológicos, ora a junção deste com um Quantizador Vetorial. Os resultados obtidos comprovam a eficácia dos meios empregados, e sugerem a sua utilização num projeto de auxílio a deficientes auditivos.

I - INTRODUÇÃO

Acredita-se, pelo conhecimento que se tem até hoje sobre sistemas nervosos biológicos, que as organizações de baixo nível de tais sistemas sejam determinadas geneticamente. Com o aprendizado, durante a vida, as organizações de níveis mais altos vão se definindo, através de um processo de **auto-organização**. Este princípio de auto-organização implica na colocação ordenada dos neurônios, de forma que possam refletir algumas características físicas do estímulo extermo.

Um algoritmo, desenvolvido por Teuvo Kohonen [6], é capaz de produzir mapas auto-organizáveis com princípios similares aos das células nervosas, cujas aplicações concentram-se principalmente nas tarefas de reconhecimento do sinal da fala. Em sua versão mais simplificada, representa um classificador não supervisionado, bastante semelhante ao algoritmo de **Quantização Vetorial** [5]. É capaz de extrair as características mais importantes dos padrões de entrada, situados num espaço multidimensional, e representá-las num sistema de poucas coordenadas.

Com estas características, esta rede revela-se bastante útil na visualização de sinais da fala num mapa bidimensional, representando-os numa forma mais genérica que os mapas formantes, pois leva em conta o espectro completo de cada quadro do sinal [4].

II - MAPAS DE KOHONEN

O processamento do sinal da fala, notadamente no caso do reconhecimento, representa um problema bastante complexo.

Trabalha-se com padrões de entradas contínuos, sem uma definição precisa do início e fim deste dentro do sinal, e a saída normalmente é definida por um símbolo.

Os mapas de Kohonen são capazes de representar, em poucas dimensões, padrões de entrada que pertencem a um espaço multidimensional. Kohonen compôs uma rede, com elementos neuronais, capaz de trabalhar com um número indefinido de padrões de entrada. Esta rede, após treinamento, pode ser utilizada para identificação de novos padrões.

Os mapas auto-organizáveis de Kohonen são constituídos basicamente de nós de entrada, uma matriz de pesos que conecta a entrada à saída, e pelos nós de saída. Tais nós de saída são arranjados numa grade, normalmente bidimensional, na forma de uma janela retangular ou hexagonal. Um exemplo de topologia desta rede pode ser visto na Figura 1.

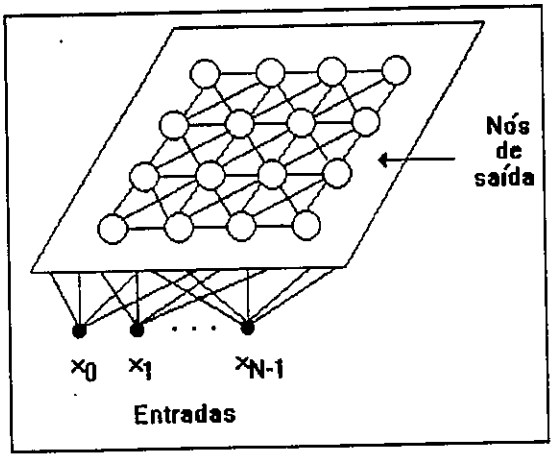


Figura 1: Mapas auto-organizáveis de Kohonen.

As entradas para a rede são contínuas, seqüenciais no tempo, e sem especificação da saída desejada, sendo, portanto, classificada como uma **rede de treinamento não supervisionado** [3]. Os nós de saída mantém-se altamente interconectados, e serão ordenados naturalmente na fase de aprendizado. A adaptação dos pesos se fará de forma que, após um número suficiente de entradas, as células da saída topologicamente próximas sejam sensíveis às entradas fisicamente semelhantes. Sendo assim, tais pesos definirão grupos que amostram o espaço de entrada. Esse processo de adaptação é também conhecido com **Treinamento Competitivo** [2]. Seu princípio básico provém de estudos de problemas estatísticos, que envolvem análise de agrupamentos.

Para que seja possível o ordenamento natural dos neurônios, o algoritmo de Kohonen requer que uma vizinhança seja definida ao redor de cada nó. O raio de tal vizinhança vai diminuindo com o tempo, conforme pode ser visto na Figura 2.

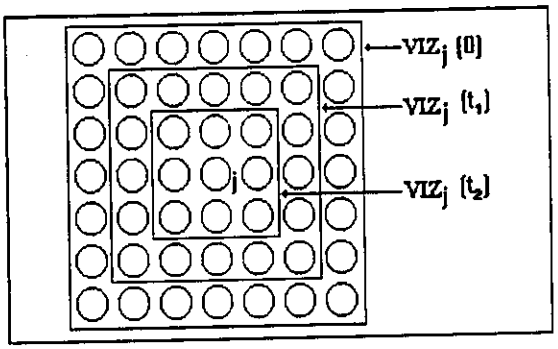


Figura 2: Exemplo de vizinhanças formadas no Mapa de Kohonen.

Também é necessário uma forma de se calcular distâncias entre nós; o que se utiliza normalmente é a Distância Euclidiana:

$$d_j = \sum_{i=0}^{N-1} (x_i(t) - w_{ij}(t))^2 \qquad (1)$$

onde

- $x_i(t)$ é a entrada do nó **i** no tempo **t**
- $w_{ij}(t)$ é o peso do nó de entrada **i** ao nó de saída **j** no tempo **t**.

Quando um nó de saída é escolhido por ter a menor distância ao nó de entrada, os pesos deste e de sua vizinhança serão modificados, usando:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(t)\,(x_i(t) - w_{ij}(t)) \quad (2)$$

$$\forall j \in VIZ_j(t) \qquad 0 \le i \le N-1$$

A função $\alpha(t)$ representa o termo de ganho ($0 < \alpha(t) < 1$), o qual decresce com o tempo t.

Esta alteração nos pesos dos nós da vizinhança é feita com a finalidade de que tais nós tornem-se mais sensíveis aquele tipo de entrada. O processo é repetido para todas as entradas. Os pesos normalmente convergirão e serão fixados após o número de iterações desejado.

Nas tarefas que envolvem processamento do sinal da fala, as entradas geralmente correspondem aos componentes espectrais da Transformada Rápida de Fourier (FFT) [7, Cap. 6], ou ainda, a coeficientes preditos linearmente (LPC) [7, Cap. 8].

III - IMPLEMENTAÇÃO

Neste trabalho, o mapa de Kohonen foi implementado para atuar como uma ferramenta que possibilitasse a visualização do sinal da fala [1]. Os sinais da fala são inicialmente preprocessados, a fim de que se possa extrair suas características. A seguir, os parâmetros obtidos são aplicados ao algoritmo de Kohonen, efetuando assim o treinamento da rede. Como último passo, para cada palavra é definida uma trajetória dentro deste mapa, a qual é composta pelos nós aos quais cada quadro mais se aproxima. Isto é feito com o intuito de verificar que trajetórias fisicamente próximas podem representar as mesmas palavras, pronunciadas em momentos diferentes.

Paralelamente ao algoritmo de Kohonen, implementou-se também o algoritmo da Quantização Vetorial [5]. Formulou-se uma maneira de trabalhar com os dois métodos em conjunto. Esta junção procede da seguinte maneira. Inicialmente, são calculados centróides, a partir dos sinais amostradas das palavras, pelo algoritmo de Quantização Vetorial. Os quadros de tais palavras passam, então, a ser representados apenas pelos centróides das células nas quais se localizam. A seguir, os centróides gerados são submetidos, como dados de entrada, ao treinamento do mapa de Kohonen. Finalmente, são calculadas e traçadas as trajetórias das palavras, cujos quadros são representados apenas pelos centróides.

Para efetuar a implementação dos algoritmos citados, e das rotinas gráficas necessárias, utilizou-se um computador da IBM, a RISC 6000/520H.

Até agora apenas foi falado sobre o trabalho com palavras isoladas. No entanto, dentro do módulo de Kohonen, ainda existe a possibilidade de se trabalhar com vogais e com frases.

IV - AQUISIÇÃO DOS DADOS

Os dados utilizados nos testes foram, em sua maioria, obtidos de arquivos contendo amostras de sinais da fala.

Para gravação dos arquivos que fizeram parte dos testes, utilizou-se um sistema de aquisição de dados, implantado num IBM PC 286. Tal sistema é composto basicamente por um filtro passa-baixa de 5 kHz, e um conversor A/D de 12 bits, com uma freqüência de amostragem de 10 kHz. Na Figura 3 encontra-se um diagrama representando o processo ao qual o sinal original é submetido.

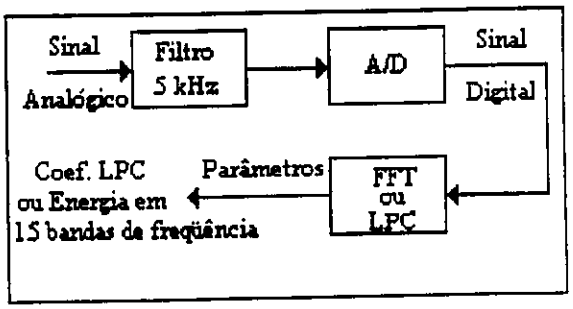


Figura 3: Diagrama do preprocessamento do sinal da fala.

O banco de dados utilizado nos testes pode ser visto na Tabela 1.

Elemento da fala	Num.amostras	Tempo gravação [s]
vogal	1.800	0,18
palavras isoladas	4.000	0,40
frases	80.000	8,00

Tabela 1: Tipos de elementos da fala utilizados nos testes.

As palavras utilizadas correspondem a 5 pronúncias isoladas de cada um dos seguintes termos:

- bola (*bola*);
- casa (*casa*);
- casa (*ccasa*);
- colo (*ccolo*);
- humano (*chumano*);
- sonhos (*csonhos*).

Entre parênteses encontram-se os nomes atribuídos aos arquivos das respectivas palavras.

V - RESULTADOS OBTIDOS

Os resultados aqui apresentados correspondem, numa primeira parte, a aplicação do algoritmo de Kohonen, e, numa segunda parte, a junção dos métodos apresentados.

Mapas de Kohonen

Os sinais aqui utilizados foram divididos em quadros de 256 amostras cada, com sobreposição de 128 amostras entre os quadros adjacentes. Para a Transformada de Fourier, usou-se 16 filtros, com 15 bandas de freqüência. Conseqüentemente, o número de nós de entrada foi mantido em 16. Ajustou-se um mapa retangular com dimensão 10x10, contendo, portanto, 100 nós de saída.

Para as vogais, a seqüência de treinamento foi composta de 50 quadros de sinal, ou seja, 5 vogais de 10 quadros cada. As trajetórias de um representante de cada vogal podem ser vistas na Figura 4. Pode-se verificar que as trajetórias geradas mantiveram-se bem distribuídas sobre o mapa, sem haver, sequer, ocorrência de sobreposição entre as mesmas.

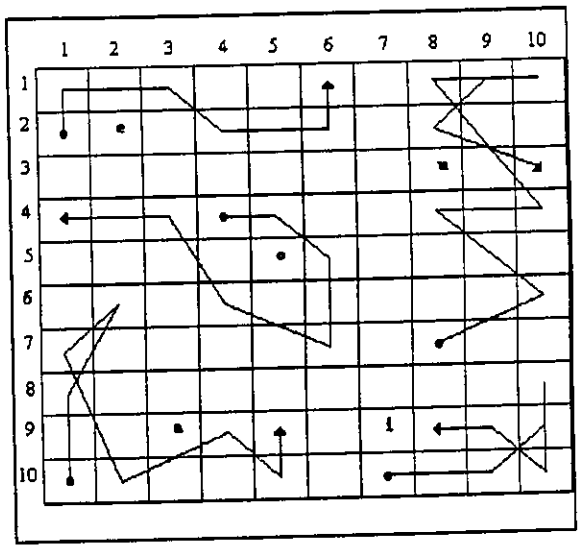


Figura 4: Trajetórias das vogais sobre o Mapa de Kohonen

Com relação as palavras isoladas, foram utilizadas as 30 palavras já citadas, sendo cada palavra composta de 25 quadros de amostras. Efetuou-se um total de 500 iterações no treinamento. As Figuras 5 e 6 mostram as trajetórias de algumas das palavras que participaram do aprendizado. Pode-se constatar novamente a propriedade dos mapas auto-organizáveis em espalhar os dados durante seu treinamento, ressaltando que, tais dados, representam os fonemas que compõem as palavras treinadas.

Como prosseguimento, foi feito o treinamento do mapa com 3 frases, sendo cada uma composta de 500 quadros. Foi efetuado um total de 100 iterações. As respectivas trajetórias correspondem a palavras contidas nestas frases, conforme pode ser visto nas Figuras 7 e 8. Não nota-se, com esse novo conjunto de treinamento, grande diferença no comportamento das trajetórias.

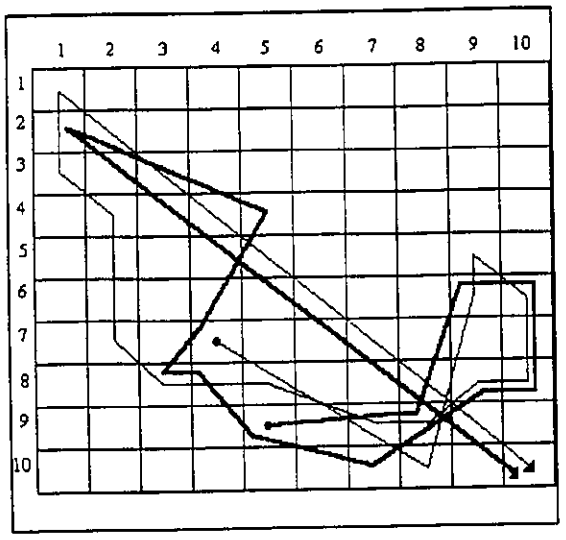


Figura 5: Trajetórias de 2 exemplares da palavra *bola* sobre os mapas auto-organizáveis treinados com 30 palavras.

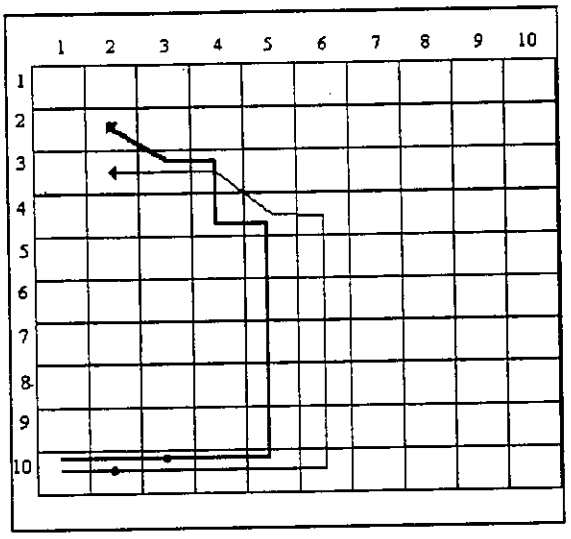


Figura 6: Trajetórias de 2 exemplares da palavra *chumano* sobre os mapas auto-organizáveis treinados com 30 palavras.

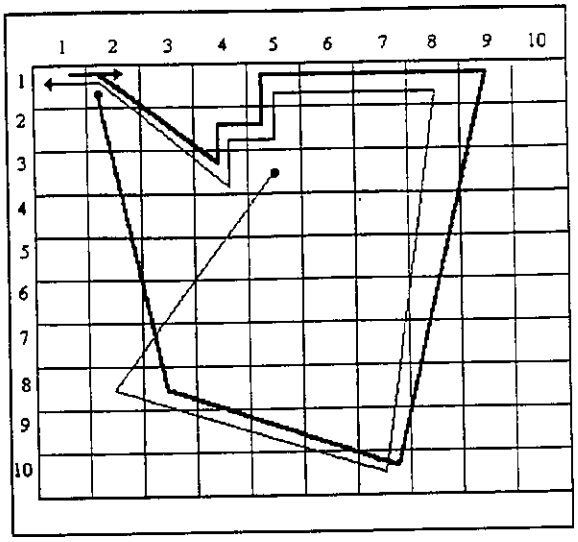


Figura 7: Trajetórias de 2 exemplares da palavra *ccasa* sobre os mapas auto-organizáveis treinados com 3 frases.

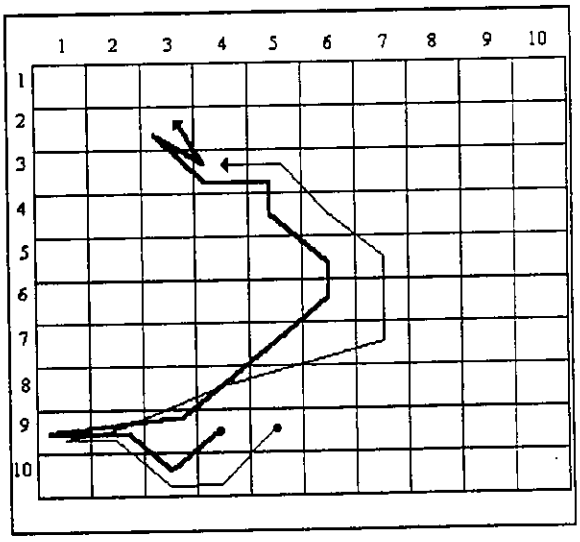


Figura 8: Trajetórias de 2 exemplares da palavra *chumano* sobre os mapas auto-organizáveis treinados com 3 frases.

Quantização Vetorial e Mapas de Kohonen

Neste módulo, as 30 palavras citadas participaram dos testes. A partir de tais palavras, foram gerados 64 centróides, com o algoritmo de Quantização Vetorial. Estes centróides formaram, então, o conjunto de treinamento do mapa de Kohonen. A dimensão do mapa continuou em 10x10 e o número de iterações foi aumentado para 1.000. Para o cálculo das trajetórias foram utilizadas as mesmas palavras, porém com seus quadros sendo representados agora apenas pelos centróides gerados. As Figuras 9 e 10 mostram alguns destes resultados. Nenhuma melhora substancial foi obtida, embora, no geral, as trajetórias tenham mantido-se bastante próximas para palavras de mesma natureza.

VI - CONCLUSÃO

Pode-se verificar, com base nos resultados obtidos, a forte propriedade do algoritmo de Kohonen em agrupar os dados de entrada, de acordo com suas

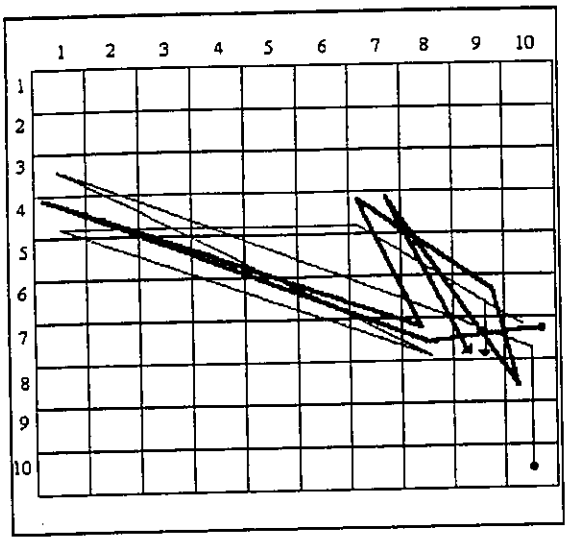


Figura 9: Trajetórias de 2 exemplares da palavra *bola* sobre os mapas auto-organizáveis treinados com 64 centróides.

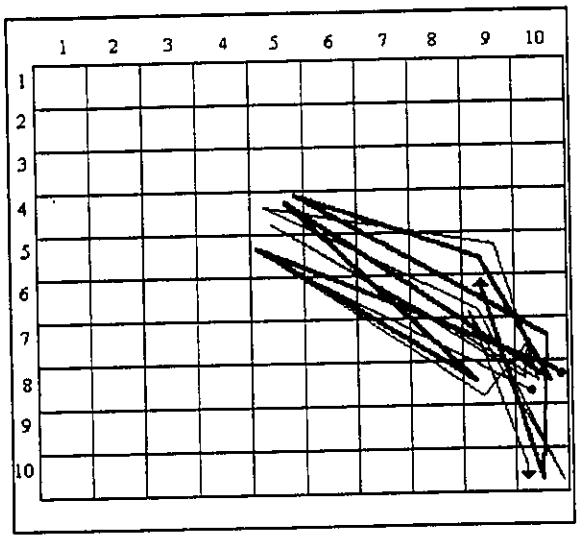


Figura 10: Trajetórias de 2 exemplares da palavra *chumano* sobre os mapas auto-organizáveis treinados com 64 centróides.

características físicas, e de espalhar estes grupos por sobre um mapa, denominado Mapa de Kohonen. Pôde-se notar, quando aplicou-se a Quantização Vetorial em conjunto com Kohonen, que essa propriedade de distribuição não se manteve de forma tão homogênea. A grande vantagem que se obteve neste caso foi em termos de velocidade, visto que o tamanho do conjunto de treinamento diminuiu consideravelmente.

Conclue-se que é totalmente viável aplicar os métodos vistos em tarefas que envolvam o reconhecimento de elementos da fala, notadamente no caso de palavras isoladas, simplesmente pela análise das trajetórias geradas. Isto é possível devido a observação de que, trajetórias fisicamente próximas, normalmente representam o mesmo elemento falado.

O algoritmo de Kohonen possibilita o desenvolvimento de um bom sistema para visualização do sinal da fala. Tal sistema poderia ser usado, por exemplo, como auxílio a deficientes auditivos, no treinamento da reprodução de palavras isoladas.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] ANTUNES, Ana C. S. *O Uso do Mapa de Kohonen para Traçado de Trajetórias do Sinal da Fala.* Uberlândia: DEENE da UFU, 1993. 135 p. (Dissertação de Mestrado).

[2] KANGAS, Jarj A.; KOHONEN, Teuvo K.; LAAKSONEN, Jorma T. Variants of self-organizing maps. *IEEE Transations on Neural Networks*, v.1, n. 1, p. 93-99, 1990.

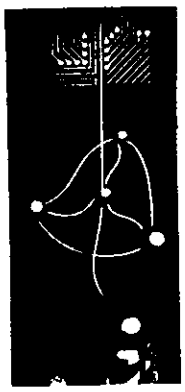
[3] KEPUSKA, Veton Z. & GOWDY, John N. Investigation of phonemic context in speech using self-organizing feature maps. *IEEE*, p. 504-507, 1989.

[4] KOHONEN, Teuvo; MÄKISARA, Kai; SARAMÄKI, Tapio. Phonotopic maps - insighful representation of phonological features for speech recognition. In: IEEE - INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION, 7., 1984, Montreal, Canada. *Proceedings*... [S.l.:S.n.], 1984. p. 182-185.

[5] LINDE, Yoseph; BUZO, Andrés; GRAY, Robert M. An algorithm for vector quantizer design. *IEEE Transaction on Communications*, v. COM-28, n. 1, p. 84-95, Jan. 1980.

[6] LIPPMANN, Richard P. An introduction to computing with neural nets. *IEEE ASSP MAGAZINE*. p. 4-23, Apr. 1987.

[7] RABINER, Lawrence R. & SCHAFER, Ronal W. *Digital processing of speech signals*. New Jersey: Prentice-Hall, Inc., 1978.

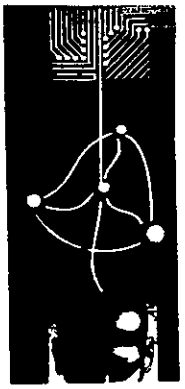


1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

PROCESSAMENTO DE SINAIS II

Anotações



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá. 24 a 27 de outubro de 1994

Soluções Conexionistas para o Reconhecimento de Irregularidades em Espectrogramas

Claudio Loesch

Universidade Regional de Blumenau
Departamento de Matemática

Resumo

O trabalho explora a pesquisa de uma solução ao problema da detecção e o reconhecimento de irregularidades em espectrogramas obtidos de regularímetros, através da implementação de uma rede híbrida caracterizada por três fases: detecção de agrupamentos para análise, processamento do mesmo e a posterior classificação do padrão.

Introdução

A regularidade dos fios têxteis é de grande importância para o aspecto final das malhas e tecidos. Na análise dos fios, o termo *irregularidade* caracteriza as variações de torção, resistência, título, aparência, etc. Estas irregularidades são geradas normalmente por órgãos defeituosos de alguma das diversas máquinas do processo produtivo, ou por regulagens defeituosas. O regularímetro serve área de manutenção mecânica, para a identificação e correção dos possíveis defeitos junto s máquinas quando detectadas irregularidades no processo produtivo da fiação.

O regularímetro apresenta, após uma análise, dois tipos de gráficos: o *diagrama de variação de massa* e o *espectrograma*. Enquanto que o primeiro registra a variação de massa do material de prova (mecha, fio ou pavio), submetido a teste, o correspondente espectrograma tem como domínio o comprimento de onda da irregularidade, discretizado em um certo número de canais, e como imagens as correspondentes amplitudes das irregularidades presentes, em cada canal. É resultante da análise harmônica do diagrama de variação de massa. Para poder abranger uma ampla faixa de comprimentos de onda dentro do domínio, usam-se comprimentos de onda por canal que crescem em escala geométrica. Nos regularímetros USTER-III os comprimentos de onda crescem na razão $\sqrt[5]{2}$. Assim, estes conseguem abranger comprimentos de onda de 1 cm 40 m com 61 canais, que é a faixa normal que cobre a maioria dos testes. Desta forma, aplica-se a escala logaritmica ao eixo das abcissas.

As fibras, componentes do material fibroso em processo, são elementos finitos aleatóriamente distribuídos. A disposição real das fibras assim obriga existência de irregularidades periódicas mesmo sob condições perfeitas de funcionamento de máquinas no processo de montagem das fibras. O espectrograma normal resulta de montagens de fibras

que podem ser realizadas técnicamente e que são livres de falhas. Assim, o material produto de cada etapa do processo produtivo possui um espectrograma normal característico, cuja forma depende ainda da máquina de onde provém. As irregularidades provocadas por defeitos mecânicos adicionam-se s irregularidades normais, e detectar a presença das primeiras é o objetivo da análise.

A constatação de uma irregularidade é simples quando se apresenta como um pico de irregularidade singular, como na figura 1. O espectrograma evidencia que o excesso da amplitude registrada em λ_0 sobre a curva aceita como normal, que, pela sua singularidade, é causado pela ocorrência de uma irregularidade senoidal de comprimento de onda λ_0.

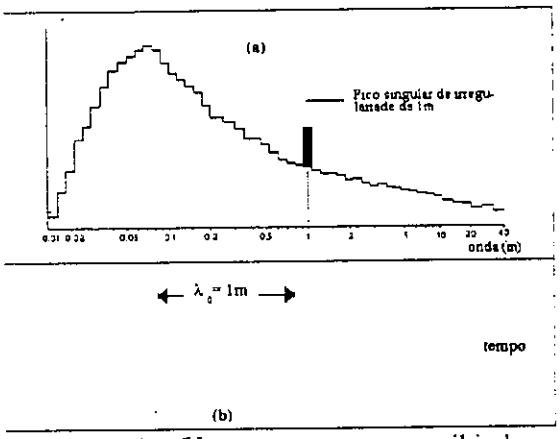


Figura 1. (a) Um espectrograma exibindo um único pico de irregularidade periódica, em um comprimento de onda λ_0. (b) Variação de tensão provocada pelo órgão defeituoso, produzindo falha periódica senoidal.

A interpretação dos diagramas torna-se mais difícil quando pretende-se descobrir variações periódicas múltiplas, que além disso, vão se sobrepondo. Ondas periódicas não-senoidais podem ocorrer. Variações assimétricas de tensão no fio, provocadas por movimento reverso na bobina do filatório podem provocar falhas periódicas como uma onda dente-de-serra (figura 2-b). O desenvolvimento desta onda, em série de Fourier, é constituído de infinitos termos de magnitudes decrescentes. A magnitude da onda fundamental, e de alguns de seus primeiros harmônicos podem ser percebidos sem muita dificuldade, como proporcionais aos excessos dos picos sobre o espectrograma normal (figura 2-a).

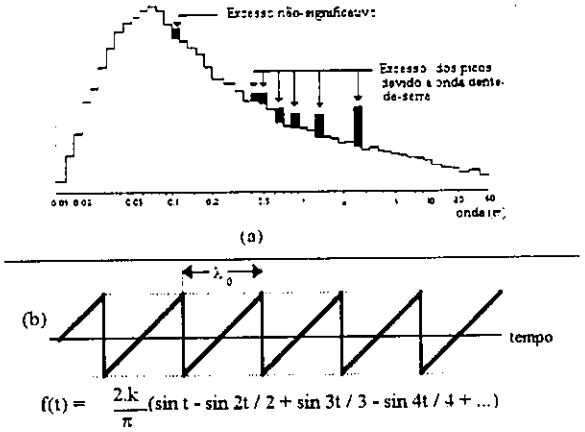


Figura 2. (a) Picos de irregularidade acima do espectrograma normal perfeitamente visíveis, todos provocados por (b) uma onda dente-de-serra de tensão assimétrica.

Dada a razão $\sqrt[5]{2}$, é facil constatar que, se o canal n contém a amplitude da irregularidade da onda da irregularidade fundamental λ_0, então seus primeiros harmônicos de comprimento de onda $\lambda_0/2$, $\lambda_0/3$, $\lambda_0/4$ e $\lambda_0/5$ contém sua amplitudes, respectivamente, nos canais n-5, n-8, n-10, e n-11,6. Neste último caso, distribuído entre n-11 e n-12.

Um pico, ao invés de possuir toda sua amplitude de irregularidade da onda concentrada no mesmo canal, pode apresentar parte distribuída para um canal vizinho. A interpretação deve levar em conta um pico único de amplitude correspondente sua soma (figura 3).

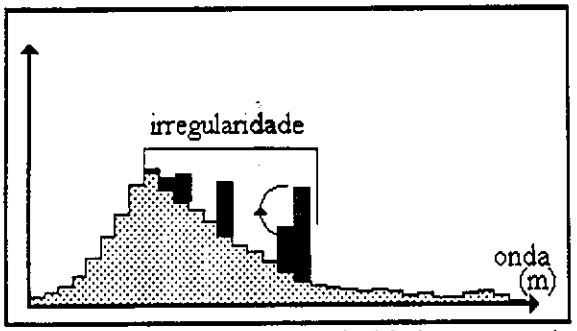


Figura 3. Em preto, a irregularidade constatada. A área sombreada indica a amplitude mais adequada ao reconhecimento, pelo deslocamento do pico do canal à esquerda.

O primeiro passo ao reconhecimento de irregularidades consiste em extrair o espectrograma normal do espectrograma em análise. O resultado, um vetor de excessos sobre o especrograma normal, é o objeto da análise. Neste, sobram: ruídos, valores não-significativos e evidências fortes de irregularidades, e este vetor será a entrada para a análise da rede.

Filtragem de Ruídos e Detecção de Agrupamentos

A segmentação da imagem, com filtragem de ruídos e detecção de agrupamentos, pode ser executada por uma rede neural cuja arquitetura consiste numa ou mais camadas dispostas sequencialmente, cada camada lateralmente conectada.

Será discutida agora a dinâmica da camada. Pelas conexões laterais, as saídas dos elementos de processamento realimentam suas próprias entradas e as entradas dos demais elementos. Nesta camada, os elementos de processamento cooperam e disputam entre si os sinais de entrada, de acordo com o sinal algébrico dos pesos das conexões, iterativamente, até alcançarem um possível estado de estabilidade. As equações do processo iterativo para cada elemento j da camada cooperativa/competitiva, na transição para a iteração (k+1) são:

$$s_j^{(k+1)} = \frac{\sum_i x_i^{(k)} . w_{ij}}{\sum_i x_i^{(k)}} + b_j \qquad (1)$$

$$x_j^{(k+1)} = F\left(s_j^{(k+1)}\right) \qquad (2)$$

onde:

$x_i^{(k)}$ é o valor do sinal do i-ésimo elemento de processamento;

w_{ij} é o peso da conexão da saida do elemento i para a entrada do elemento j;

$s_j^{(k+1)}$ é o sinal integrado do elemento j;

b_j é um escalar;

F é a função de transferência.

Para precaver contra crescimentos de magnitudes de sinais de forma desordenada e não-limitada, escolhem-se para F funções que prendem os limites superiores e inferiores de seus argumentos. Uma solução simples consiste em adotar

$$F(x) = \begin{cases} 0 \text{ se } x < 0 \\ 1 \text{ se } x > 1 \\ x \text{ se } 0 \leq x \leq 1 \end{cases} \qquad (3)$$

Escolhendo adequadamente os pesos das conexões, a convergência de (3) fica assegurada. Devem-se escolher pesos que refletem característica de distribuição espacial e características comuns a todos os padrões possíveis de serem classificados. Se os padrões são conexos no domínio, escolhem-se pesos positivos para as conexões de cada elemento com seu espacialmente vizinho e, eventualmente, para os próximos vizinhos. No entanto, pesos negativos, de efeito inibitório, devem encontrar-se presentes em **W**, para que exista efeito competitivo, pois caso contrário, o agrupamento seria global.

Desta forma, valores maiores tendem se fortalecer ainda mais, e inibir outros

eventuais agrupamentos sendo formados. De acordo com a força inibitória, mais de um agrupamento pode sobreviver na estabilização, ou pode sobreviver um único agrupamento. Em qualquer circunstância, a convergência deverá ocorrer para um vetor valores binários 1/0. Elementos que estabilizam com unidades formarão agrupamentos para análise. Após a convergência, processa-se o produto entre os elementos do vetor de entrada e seus correspondentes elementos no vetor de saída. O resultado deverá ser um vetor contendo um único padrão de irregularidade isolado. A maioria dos ruídos fracos terão também sido eliminados pelo processo competitivo. A escolha de valores b_j negativos fará com que, na ausência de valores de entrada significativos, nenhum sobreviva.

Para o problema de reconhecimento em questão, foi utilizado o conjunto de pesos calculados por (4). Estes valores diminuem, de acordo com o aumento da distância, passando ao efeito competitivo quando $|i-j| \geq 6$.

$$w_{ij} = 3.5\left(e^{-0.2|i-j|} - 0.25\right), \quad b_j = -0.05 \qquad (4)$$

Pode ocorrer, no entanto, que alguns valores não-pertinentes à irregularidade façam também parte do agrupamento. Caso ocorra sobreposição de dois ou mais padrões, ligeiramente defasados em localização, agrupamentos mais largos abrangendo sua união serão criadas. No caso de ausência de qualquer irregularidade de intensidade significatica, acima do espectrograma normal, nenhum agrupamento sobrevive.

Incluiu-se uma nova camada mais competitiva, que recebe os dados que passam pela janela de tamanho N criada pela etapa anterior. Nesta, utilizou-se pesos de acordo com (5), para uma filtragem mais restritiva.

$$w_{ij} = \begin{cases} 2.2 \text{ se } i = j \\ -1.15/N \text{ se } i \neq j \end{cases} \qquad (5)$$

Sobrevivem os sinais mais fortes das ondas de estiragem (em torno de 8), e nos casos de ondas com harmônicos, até 5 ou 6 picos pertinentes mais fortes, suficientes para bem caracterizar a onda. Os sinais fora do grupo de canais significativos, que contém apenas ruídos, desaparecem No caso de pico único, eventualmente podem passar uns poucos outros canais (constatado 2 a 3, em média) além deste. Desta maneira, é possível selecionar um agrupamento de forma mais adequada ao tratamento posterior.

Redes de Propagação para Frente e Aprendizado Backpropagation

As redes de propagação para frente, com múltiplas camadas, e pesos de conexões ajustáveis, comumente aprendizagem via Backpropagation, tem sido apontadas pela literatura como um potente instrumento para classificação de padrões. A prática tem demonstrado poucos resultados animadores em relação ao treinamento simultâneo de todas estas características, utilizando Backpropagation somente. Como alternativa, redes híbridas acenam com possibilidades de utilizar diversos paradigmas combinados, desta forma podendo ser conseguida uma performance melhorada. Isto é conseguido incluindo ou alterando características de topologia, dinâmica e/ou aprendizagem na rede.

Pao sugere a introdução de redes com "functional link", como uma forma alternativa equivalente ao trabalho desempenhado pelas camadas ocultas da rede: o de abstrair características de pa-

drões. A mais compelente razão para a escolha deste modelo é a do conhecimento a priori de certas transformações funcionais em que os dados irão expor suas características salientes, o que pode reduzir a rede pela eliminação da camada oculta. A técnica consiste em efetuar um pré-processamento nos dados de entrada, para então aplicar os dados transformados à rede. Como desvantagem, aponta-se para o fato de que não é claro o modo de identificar as funções "functional link" adequadas a um problema específico.

Spirkovska e Reid aperfeiçoaram uma arquitetura de redes neurais de ordem mais elevada, as redes pi-sigma (ou redes HONNs) cujo pré-processamento consiste de todas as somas de produtos de elementos distintos da camada de entrada. Trata-se de uma rede de ligações funcionais especificamente projetada para problemas de reconhecimento de padrões bidimensionais, independente de escala, rotação e translação. Restringindo a ordem dos produtos ao máximo de três, obtiveram sucesso em classificar padrões binários com 100% de acerto, em quadros de pixeis binários de dimensão até 128X128.

Embora o vetor de entrada do problema presente seja unidimensional, com valores reais, seria teoricamente possível tentar considerar o reconhecimento do gráfico da irregularidade, desde que se discretize a escala vertical. Porém, para boa confiabilidade, a discretização deve ser ampla o suficiente para discernir entre diferentes possíveis padrões e difentes amplitudes de irregularidades com os quais os padrões podem ocorrer. Ocorre que a complexidade computacional depende do número de possíveis combinações 3 a 3, muito sujeito à explosão combinatória. Em virtude da acuracidade necessária às amplitudes das irregularidades, a possibilidade de uso desta idéia não foi testada.

Processamento do Agrupamento Selecionado

Admita-se que o agrupamento resultante das duas camadas cooperativas/competitivas tenha tamanho n. A próxima etapa é tratar do problema de invariância a mudanças de escala e a translações da irregularidade contida no espectrograma. Ambos podem ser conseguidos através de processamento intermediário.

A invariância à mudança de escala pode ser efetivada, calculando-se o vetor **x** a partir da normalização do vetor **y** do agrupamento criado pela segunda camada cooperativa /competitiva.

$$\mathbf{x} = \mathbf{y} / \|\mathbf{y}\| \qquad (6)$$

A transformada de Fourier preserva as propriedades fundamentais dos dados, o que a torna unívocamente definida. Além disso, ela pode ser discretizada, o que torna viável seu uso computacional. Dado um vetor $\mathbf{x} = (x_0, x_1, \ldots, x_{N-1})$, sua transformada de Fourier discreta resulta num vetor complexo **X**, cujas componentes podem ser calculadas por (7), ou por métodos mais eficientes, como a transformada rápida de Fourier.

$$X_i = \frac{1}{N} \sum_{k=0}^{N-1} x_k e^{-j2\pi nk/N}; \; 0 \le i \le N-1 \qquad (7)$$

A sua propriedade mais importante, refere-se ao deslocamento. Para uma translação inteira h, o vetor $\mathbf{X}'$ de componentes $x'_k = x_{k-h}$ tem por transformada

$$X'_n = X_n . e^{-j2\pi nh/N} \qquad (8)$$

Desta forma, $|X_i'| = |X_i|$, para todo i , e a amplitude da transformada é invariante à translação, o que assegura a invariância à translação. Estes dois processamentos intermediários conferem à rede de propagação para frente uma característica "functional link".

Classificação do Padrão

No problema de classificação, é importante sabermos qual padrão de irregularidade que foi detectado, e também o respectivo comprimento de onda. Podemos visualizar duas soluções ao problema de classificação. Na primeira solução, implementamos as duas camadas cooperativas/competitivas como descrito. Posteriormente, é feita a classificação do vetor de dados normalizados do agrupamento sobrevivente, em uma rede de alimentação para frente. Neste caso, sem necessidade de nenhuma transformada de Fourier intermediária, pois apenas o agrupamento sobrevivente é enfocado, dentro do qual não haveria nenhum problema de translação. O padrão classificado com sucesso, mais a localização do agrupamento deveriam permitir a localização da onda fundamental, ou, se for uma onda de estiragem, seu ponto de centro.

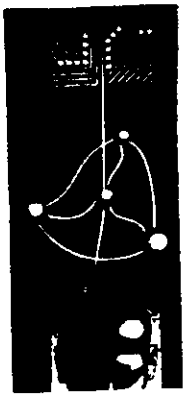
A segunda solução consiste também nas duas camadas cooperativas/competitivas e normalização do vetor do agrupamento vencedor de saída, mas seguida por uma transformação de Fourier discreta sobre o vetor de domínio do espectro, contendo o agrupamento sobrevivente e demais valores nulos. Posteriormente, é feita a classificação numa rede de alimentação para frente, treinada para reconhecer as transformadas dos possíveis padrões normalizados. A localização do pico da onda fundamental, necessário para auxiliar na pesquisa do órgão defeituoso, pode se obtida a partir de (8). Se a i-ésima componente de **X** e de **X'** possuem ângulos de fase φ_{X_i} e $\varphi_{X_i'}$, respectivamente, então

$$h = \left(\varphi_{X_i} - \varphi_{X_i'}\right) N / (2\pi i) \qquad (9)$$

permite obter h a partir de qualquer valor de i. Porém, supondo que **X'** seja a transformada de Fourier do padrão **x'** normalizado e correto, dentro do vetor de tamanho n, e que **X** seja a transformada a ser classificada pela rede, contendo a irregularidade correspondente ao mesmo padrão, o cálculo da mão-direita de (9) poderá fornecer resultados diferentes para diferentes valores de n, em virtude de ruídos e imprecisões diversas. A prática demonstrou que a média destes valores não é um bom estimador para h, pois para alguns poucos valores de i, os ângulos de fase φ_{X_n} e $\varphi_{X_n'}$ podem diferir significativamente. Não obstante, a mediana tem demonstrado bons resultados.

Conclusão

Qual solução adotar? A primeira solução, mais simples, evita a necessidade de computar a transformada de Fourier e opera com uma rede posterior de propagação para frente mais simples. Isto reduz sensivelmente tempo e esforços computacionais. Porém, o entrave consiste no fato de que não há garantia sobre o tamanho da janela proveniente das camadas cooperativas/competitivas. Embora isto não comprometa a posterior classificação do padrão, o problema é a impossibilidade de conseguir, por este meio, assegurar a exata localização da onda fundamental dentro do espectrograma. Esta localização é fundamental para determinar o comprimento de onda, e posteriormente localizar a causa.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajuba
Itajuba. 24 a 27 de outubro de 1994

Investigação de Aproximadores de Funções Através de Interpolação, Transformadas Ortogonais e Redes Neurais

João Fernando Marar[1,2]
Edson Costa de Barros Carvalho Filho[2]

[1]UNESP–Universidade Estadual Paulista
Faculdade de Ciências - Departamento de Computação
Bauru - São Paulo - Brasil

[2]UFPE–Universidade Federal de Pernambuco
Cx 7851. 50.732-970. Recife. PE. Brasil
{jfm,ecdbcf}@di.ufpe.br

Sumário. Atualmente. muitos trabalhos têm investigado a utilização de redes neurais multicamadas feedforward como aproximadores universais de funções. Neste artigo abordaremos as técnicas mais usuais em aproximadores de funções. incluindo interpolação. transformadas ortogonais e redes neurais multicamadas.

1 Introdução

Devido às dificuldades na manipulação de certas funções importantes que aparecem em problemas associados a *matemática aplicada*. torna-se necessária sua representação aproximada através de funções mais simples. de maneira que. estas últimas substituam as primeiras. O mesmo procedimento é utilizado quando é necessário trabalhar com funções cuja expressão analítica é desconhecida. *i.e.*. funções as quais se conhece somente alguns de seus valores através de determinações experimentais. Em geral. esta funções aparecem no cotidiano científico sob a forma de sequência de valores (tabelas). Nestes casos. estão inclusos problemas associados a *processamento de sinais: processamento e reconhecimento padrões entre outros*. uma das áreas de grandes aplicações destas técnicas. A utilização dos computadores digitais para o tratamento numérico destes problemas é inevitável. dada a grande massa de dados envolvida e a quantidade enorme de processamento aritmético exigido pelos algoritmos. para esta finalidade. Por exemplo. em síntese de instrumentos de orquestra. uma amostra de 1 segundo de um tom musical. requer acima de 40.000 valores reais [MAR 92].

Quando uma tabela é obtida por meio de uma expressão analítica ou determinada experimentalmente: esta função pode ser representada aproximadamente pela equação 1. por simplicidade estamos nos referindo a funções de uma única variável. contudo o raciocínio se extende para funções de várias variáveis.

$$f(x) \simeq v(x) = \sum_k a_k u_k(x) \qquad (1)$$

onde a_k são constantes e u_k podem ser funções tais como polinômiais. exponenciais. trigonométricas ou conjuntos de funções básicas ortogonais.

Caminhando de encontro ao exposto. este trabalho aborda algumas técnicas matemáticas utilizadas como aproximadores de funções. tais como a interpolação. transformadas ortogonais e por fim a apresentação de redes neurais que também são utilizada para este propósito. Este último tópico tem sido muito explorado por uma grande quantidade de pesquisadores. cujo objetivo principal é encontrar meios de formalizar problemas em redes neurais. de maneira os resultados obtidos venham a facilitar as futuras implementações [FUN 89]. [HEC 89]. [HOR 89].[HAR 90].[LES 93].[MAR 94].

2 Interpolação

A interpolação é uma técnica muito utilizada quando somente possuímos uma tabela com valores de uma certa função para um conjunto de argumentos e, é necessário o cálculo, suficientemente preciso, de algum valor não tabelado, mas que é intermediário a valores consecutivos da mesma.

Como ocorre frequentemente em vários problemas experimentais, deseja-se o valor da função correspondente a um argumento que não figura na tabela nem é intermediário a valores consecutivos da mesma. Neste caso, é necessário fazer uma extrapolação a fim de aproximar o valor desejado. A extrapolação é menos precisa que a interpolação. Mas, ambas constituem, essencialmente, um único algoritmo de aproximação.

De maneira compacta, vamos definir intuitivamente o problema da interpolação. Suponha que seja fornecido uma sequência de valores, correspondentes a uma dada função. Nosso desejo é obter os valores de $f(\bar{x})$, onde $\bar{x}$ é um valor não tabelado. Para isso construímos, a partir desta sequência de valores, uma nova função $\psi(\bullet)$ que interpola $f(\bullet)$, tal que:

$$\forall x_i, x_0 \leq x_i \leq x_n \Rightarrow \psi(x_i) = f(x_i)$$

$$\forall x \in [x_0, x_n] \Rightarrow \psi(x) \cong f(x)$$

A função $\psi(\bullet)$ que interpola $f(\bullet)$ geralmente pertence a uma família de funções, dentre as mais importantes destacamos: as polinômiais, as exponenciais e as trigonométricas.

Um resultado de grande importância para a teoria da aproximação é o teorema de *Stone-Weierstrass*, que diz:

Se $f(x)$ é contínua em $[a, b]$, então para qualquer $\varepsilon > 0$ existe um polinômio $p_n(x)$ de grau n, tal que $\mid f(x) - p_n(x) \mid < \varepsilon$ para $a \leq x \leq b$.

A prova deste teorema pode ser vista em Handscomb [HAN 65]. Este teorema constitue uma boa razão para o grande uso dos polinômios, pois estamos interessados em que a aproximação para $f(\bar{x})$, onde $\bar{x}$ não está tabelado, seja suficientemente exata. Embora a prova deste teorema seja construtiva, o polinômio resultante costuma ser de grau elevado, de modo que tal polinômio passa a não ser prático. Além disso, o teorema não diz nada sobre a existência de um polinômio interpolante para um conjunto de dados $\{(x_0, y_0), (x_1, y_1), ..., (x_{n-1}, y_{n-1})\}$, qualquer.

Mesmo que decidamos interpolar n pontos pela abordagem polinomial, nem sempre é melhor usar o polinômio global, *i.e.*, de grau $n - 1$. Um bom exemplo dos problemas associados a divergência que ocorrem com interpolação polinômial pode ser visto através da *função de Runge* [ISA 66], [STE 50], [CLÁ 89].

3 Transformadas Ortogonais

As transformadas ortogonais constituem uma poderosa ferramenta matemática para representar funções extremamente complexas através de outras mais simples. Uma importante aplicação desta técnica é a redução de dimensionalidade ou compressão de dados [AHM 75],[WOM 77]. Estas transformadas podem ser divididas em duas classes, segundo as funções básicas utilizadas. Desta forma, temos a classe baseada em funções não senoidais e a de funções senoidais, tendo esta última como única representante a transformada de Fourier.

Em geral, aplica-se esta técnica a um conjunto de dados obtidos experimentalmente, cujo objetivo é encontrar uma lei de formação para a função. Os motivos que nos levam a não usarmos a interpolação são dois, em especial: (1) Devido a erros experimentais no conjunto de dados amostrados, não faz sentido calcular exatamente a função que origina os pontos e por isso, em vez de procurar uma função que interpola os n pontos dados, procuramos uma representação que melhor se ajuste aos pontos dados. Na realidade, o ajustamento traduz um comportamento médio e (2) para um conjunto com muitos elementos, a solução da aproximação dos dados por interpolação pode consumir muita memória e ou tempo de processamento.

Detalhes sobre a representação de funções através desta técnica serão expostas a seguir, bem como apresentaremos a transformada Karhunen-Loève, como uma das inúmeras representante da classe das transformadas ortogonais que são definidas através funções básicas não senoidais.

3.1 Aproximação de Funções via Transformadas Ortogonais

Seja $\Psi = \{w_0(t), w_1(t), ..., \psi_n(t), ...\}$ um conjunto de funções reais e contínuas,(utilizamos funções reais por conveniência), será dito ortogonal no intervalo $(t_0, t_0 + T)$ se:

$$\int_{t_0}^{t_0+T} \psi_m(t)\psi_n(t)dt = \begin{cases} c_n & \text{se } m = n \\ 0 & \text{se } m \neq n \end{cases} \quad (2)$$

Para o caso onde $c_n = 1$, o conjunto Ψ é chamado ortonormal. Seja $x(t)$ uma função de valores reais, definida em um intervalo $(t_0, t_0 + T)$, e suponha que $x(t)$ possa ser escrita na forma :

$$x(t) = \sum_{n=0}^{\infty} a_n \psi_n(t) \quad (3)$$

então os coeficientes a_n podem ser obtidos da seguinte forma : multiplicamos ambos os lados da equação 3.2 por ψ_m e integramos o resultado no intervalo $(t_0, t_0 + T)$, onde obteremos :

$$\int_{t_0}^{t_0+T} x(t)\psi_m(t)dt = \int_{t_0}^{t_0+T} \sum_{n=0}^{\infty} a_n \psi_n(t)\psi_m(t)dt$$

Como ψ_m e ψ_n são ortogonais, temos :

$$a_m = \frac{1}{c_n} \int_{t_0}^{t_0+T} x(t)\psi_m(t)dt, \; m = 0, 1, 2, ...$$

Um conjunto de funções ortogonais é chamado fechado ou completo se for verificada a seguinte condição :

Para qualquer parte contínua de $x(t)$ com :

$$\int_{t_0}^{t_0+T} x^2(t)dt < \infty$$

Qualquer que seja $\epsilon > 0$, existe N tal que seja possível representar uma aproximação de $x(t)$ por uma expansão finita :

$$\int_{t_0}^{t_0+T} | x(t) - \hat{x}(t) |^2 \, dt < \epsilon$$

onde :

$$\hat{x}(t) = \sum_{n=0}^{N-1} a_n \psi_n(t)$$

Pelo desenvolvimento acima, é visível que por uma expansão em funções ortogonais, sempre será possível representar $x(t)$ por um conjunto infinito,mas enumerável $\{a_0, a_1, a_2, ...\}$. Entretanto, quando Ψ for completo torna-se possível uma aproximação de $x(t)$ através de um conjunto finito $\{a_0, a_1, ..., a_{N-1}\}$.

3.2 Aproximação de Funções via Transformadas Karhunen-Loève

Seja $\{X\}$ um conjunto de vetores, obtidos por amostragem. Um representante de $\{X\}$ é dado por $x_j = (x_{j,1}, ... x_{j,K})$. A amostra x_j pode ser aproximada por 4:

$$x_j = y_{j,1}\psi_1 + y_{j,2}\psi_2 + ... + y_{j,N}\psi_N = \sum_{i=1}^{N} y_{j,i}\psi_i \; N < K \quad (4)$$

$$y_{j,i} = x_j^t \psi_i \quad i = 1, 2, ... N \quad (5)$$

onde K é o número total de componentes da amostra e N é o número de componentes utilizadas na aproximação.

Por definição, o mínimo erro quadrado, ε, é dado pela expressão 6:

$$\varepsilon = (\sum_{i=1}^{K} y_{j,i}\psi_i - \sum_{i=1}^{N} y_{j,i}\psi_i)^2 = \sum_{i=N+1}^{K} \psi_i^t R_X \psi_i \quad (6)$$

onde R_X é a matriz de covariância do conjunto $\{X\}$.Dada por $R_x = \frac{1}{V}\sum_{j=1}^{V}(x_j - \bar{X})(x_j - \bar{X})^t$, onde V representa o número total de elementos do conjunto $\{X\}$ e $\bar{X}$ é o vetor médio do referido conjunto.

Quando $\{\psi_i\}$ constituem a base ortogonal de Karhunen-Loève, os elementos ψ_i são determinados a partir dos autovetores de R_X, de acordo com a equação 7:

$$R_X\psi_i = \lambda_i\psi_i \quad (7)$$

O erro de truncamento da equação 6 é minimizado pela equação 8

$$Min(\varepsilon) = \sum_{i=N+1}^{K} \lambda_i \quad (8)$$

Isto significa que, se utilizarmos apenas N auto-vetores para a representação de funções, o erro de truncamento será mínimo, sendo dado pela equação 8. A equação 4, escrita em termos

dos auto-vetores da matriz de covariância, é denominada expansão Karhunen- Loève. A correspondente transformação ortogonal inversa, na equação 5, é chamada transformada Karhunen-Loève (K-L) [CHE 91].

4 Aproximação de Funções via Redes Neurais

Em 1969, Minsky e Papert demonstraram que as redes percepton com uma camada de neurônios com funções de ativação lineares ("*Single Layred Perceptron*"), são incapazes de aproximar quaisquer funções, no máximo elas poderiam mapear uma classe muito pequena de funções especiais, as linearmente separáveis [MIN 69]. Na época, Minsky e Papert sabiam que as redes multicamadas, com arquitetura feedforward eram capazes de proporcionar uma discriminação de funções não lineares, contudo, ainda era desconhecido um algoritmo de aprendizagem eficiente para estas redes. Efetivamente, as pesquisas com redes multicamadas foram reiniciadas nos meados da década de 80, mostrando que estas redes com a arquitetura feedforward podem ser utilizadas como aproximadores universais de funções [HOR 89], [HEC 89], definindo um mapeamento num espaço de dimensão finita para outro, através do algoritmo *Backpropagation* desenvolvido por Rumelhart et.al. [RUM 86].

As redes multicamadas perceptron, conhecidas por *MLP*, são sem dúvida nehuma o tipo mais popular de redes neurais e que podem ser aplicadas em aproximadores de funções. Os algoritmos de aprendizagem para este tipo de rede foram desenvolvidos independentemente por Parker em 1985 e Rumelhart et.al. [RUM 86]. Uma formulação geral, para as redes MLP com apenas uma camada escondida e uma saída, é dada pela equação 9:

$$f(\vec{x}) = h(\sum_{i=1}^{k} c'_i h(\sum_{j=1}^{d} c_{i,j}\, x_j + c_{i,d+1}) + c'_{k+1}) \quad (9)$$

onde $\vec{x}$ é um vetor d-dimensional, correspondente a entrada, $h(\bullet)$ deve ser uma função suave (diferenciável), monotônicamente crescente, c_s e c'_s são coeficientes, k é o número de neurônios escondidos e d é a quantidade de elementos da entrada.

O algoritmo backpropagation é baseado no método do gradiente descendente [RUM 86],[HER 91], [HEC 89], o qual não garante chegar sempre ao erro global mínimo. Contudo, na grande maioria das vezes os resultados apresentados pelo algoritmo são bem próximos aos desejados.

Na prática, as funções de ativação utilizadas para as redes multicamadas são as funções sigmoides [FUN 89],[HEC 89],[SNA 90], tangentes hiperbólicas, gaussianas [HAR 90] e as funções básicas radiais [PAR 91],[SAH 90],[KAD 90], onde a quantidade de neurônios nas camadas escondidas são incrementados até que a precisão desejada seja alcançada na aproximação. Na realidade, a função complexa que uma rede *MLP* aproxima pode ser vista como composições e combinações das funções de ativação. Inclusive, a própria função analítica pode ser escrita, verificando a arquitetura da rede.

Os trabalhos [CUN 87] e [LAP 88], mostram que para adequar a aproximação de uma função desconhecida, utilizando funções do tipo *squashing* é necessário apenas duas camadas. Gallante e White [GAL 88], mostraram que uma particular rede feedforward, com apenas uma camada de funções cossenoides *squashing* é capaz de realizar a *transformada de Fourier*, a qual constrói a aproximação em série de Fourier para uma dada função. Esta rede possue todas as propriedades da representação em séries de Fourier, em particular, são apenas capazes de representar qualquer função quadrado integrável, sobre um conjunto compacto de funções, usando um número finito de unidades escondidas.

Hornik [HOR 89], faz uso do teorema de *Stone-Weirestrass* e as funções cossenos *squashing* de Gallant e White para estabelecer um padrão nas arquiteturas de redes feedforward. Hornik, usando funções *squashing* arbitrárias garante que estas podem aproximar qualquer função com qualquer grau de precisão, desde que exista a quantidade necessária de unidades escondidas disponíveis. Mas não faz mensão nenhuma sobre a quantidade de unidades necessárias para uma dada aproximação. Leshno [LES 93], garante, na mesma linha de raciocínio de Hornik que as funções básicas utilizadas como função de ativação nas redes multicamadas não podem ser polinômios, para que tais

redes sejam aproximadores universais de funções.

5 Conclusões

Neste trabalho, apresentamos algumas técnicas mais usualmente utilizadas em aproximação de funções, entre as quais destacamos a interpolação, transformadas ortogonais e redes neurais.

As redes neurais tem mostrado ser uma técnica bastante atraente para a solução de diversos problemas, dentre os quais destacam-se o reconhecimento de padrões, aproximação de funções, entre outros. Embora ainda existam limitações para o seu uso, devido a falta de um formalismo na especificação e na análise dos modelos de redes neurais, *i.e* , a determinação de um modelo adequado para cada tipo de problema, uma vez que para se compreender os mecanismos fundamentais das redes é necessário realizar simulações que na maioria dos casos são tarefas árduas e distantes da realidade do modelo. Essa dificuldade se dá em virtude da complexidade da análise matemática envolvida: o que corresponde em redes neurais a complexidade dos mecanismos de aprendizagem, a estabilidade e a topologia adequada da rede. Esta complexidade matemática inviabiliza decidir qual o modelo mais adequado.

6 Agradecimentos

Ao Departamento de Computação da UNESP-Bauru, pelo afastamento concedido para a realização do programa de doutoramento no DI/UFPE.

À Capes-PICD e CNPq pelo apoio financeiro.

References

[AHM 75] N. Ahmed, K.R. Rao. *Ortogonal Transforms for Digital Signal Processing*, Springer-Verlang, New York, 1.975.

[CHE 91] C. S. Chen, K. S. Huo. *Karhunen-Loève Method for Data Compression and Speech Synthesis*, IEE Proceedings-I, Vol 138.Nro 5, October 1991, p 377-380.

[CLÁ 89] D. M. Cláudio, J. M. Marins, *Cálculo Numérico Computacional.* Editora Atlas S.A., São Paulo, 1.989.

[CUN 87] Y. L. Cun. *Modeles connexionistes de l'apprentissage*, These de Doctorat, Universite Pierre at Marie Curie, 1.987.

[FUN 89] K. Funahashi, *On the Approximate Realization of Continuos Mappings by Neural Networks*, Neural Networks, vol 2, 1.989, p 183-192.

[GAL 88] A. R. Gallant , H. White, *There exists a neural network that does not make avoidable mistables*, In IEEE Second International Conference on Neural Networks, San Diego, 1.988, p 657-664.

[HAN 65] D. C. Handscomb *Methods of Numerical Approximation*, Pergamon Press, Oxford, 1.965.

[HAR 90] E. J. Hartman, J. M. Keeler, *Layered Neural Networks with Gaussian Hidden Units as Universal Approximations*, Neural Computation 2, 1.990, p 210-215.

[HEC 89] R. Hecht-Nielsen *Theory of the Backpropagation Neural Network*, IJCNN 89, june, 1.989, p 593-605.

[HER 91] J. Hertz , A. Krogh , R. G. Palmer *Introduction to the Theory of Neural Computation*, Addison-Wesley Publishing Company, 1.991.

[HOR 89] K. Hornik, *Multilayer Feedforward Networks are Universal Approximators*, Neural Networks, vol 2, 1.989, p 359-366.

[ISA 66] E. Isaacson, H. B. Keller, *Analysis of Numerical Methods*, John Wiley & Sons, Inc. New York, 1.966.

[KAD 90] V. Kadirkamanathan et.al. *Sequential Adapitation of Radial Basis Function Neural Networks and its application to Time-series Prediction*,

[LAP 88] A. Lapedes, R. Faber *How neural networks work*, Tech. Report LA-UR-88-418, Los Alamos National Laboratory, 1.988.

[LES 93] M. Leshno et. al. *Multilayer Feedforward Networks with a Nonpolynomial*

Activation Function can Approximate any Function, Neural Networks, vol 6., 1.993, p 861-867.

[MAR 92] J. F. Marar *Utilização da Transformada Karhunen-Loève em Síntese de Tons Musicais*, Dissertação de Mestrado -USP-São Carlos, 1.992.

[MAR 94] J. F. Marar, E.C.d.B.C. Filho *Aproximadores de Funções* , Relatório Técnico, UFPE-DI, Março,1994.

[MIN 69] M. Minsky , S. Papert *Perceptrons*, MIT Press, 1.969.

[PAR 91] J. Park.I. W. Sandberg *Universal Approximation Using Radial Basis Function*, Neural Computation, nro 3, 1.991, p 246-257.

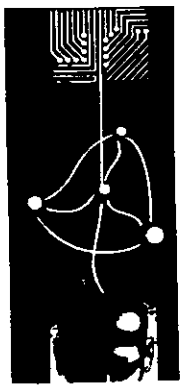
[RUM 86] D. E. Rumelhart , J. L. McClelland *Parallel Distributed Processing-Vol 1: Foundations*,The MIT Press, Cambridge,1986.

[SAH 90] A. Saha *Oriented Non-Radial Basis Functions for Image Coding and Analysis*, 1.990.

[SNA 90] R. R. Snapp et.al. *Generalizing Smoothness Constraints From Discrete Samples*, Neural Computation 2, 1.990, p 188-197.

[STE 50] J. F. Steffensen *Interpolation*, Chelsea Publishing Company, New York , 1.950.

[WOM 77] M. E. Womble, J. S. Halliday, S. K. Mitter, M. C. Lancaster, J. H. Triebwasser *Data Compression for Storing and Transmitting*, Proceedings of the IEEE, vol 65, Nro 5, May 1977, p 702-706.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

Soluções Conexionistas Híbridas para o Reconhecimento de Padrões Unidimensionais e Bi-dimensionais

Solange Sari
Claudio Loesch
Ricardo Miranda Barcia

- Professora do Departamento de Computação na UNISUL - Universidade do Sul de Santa Catarina (Tubarão - SC). e-mail: eps3ssa@brufsc.bitnet.
- Professor do Departamento de Matemática na FURB - Universidade Regional de Blumenau (Blumenau - SC). e-mail : furb@brufsc.bitnet.
- Professor do Curso de Pós-Graduação em Engenharia de Produção na UFSC - Universidade Federal de Santa Catarina (Florianópolis - SC). e-mail: eps1rmb@brufsc.bitnet.

Resumo

Um dos problemas mais genéricos de reconhecimento de padrões envolve a situação em que um ou mais padrões podem simultaneamente encontrar-se contidos dentro de um domínio discreto, gradeado e limitado de $\Re$ ou de $\Re^2$. Este trabalho considera o reconhecimento independente das características de translação, escala ou rotação no plano. A abordagem combinatorial, torna-se impraticável, mesmo para problemas de pequeno porte devido à explosão combinatória de possibilidades geradas por estas características. Propõe-se explorar a solução do problema através da implementação de uma rede neural híbrida caracterizada por três fases de transformação de dados: segmentação da imagem (com filtragem de ruídos e detecção de agrupamento), processamento do agrupamento enfocado e o posterior reconhecimento do padrão.

Introdução

As redes de alimentação para frente, com múltiplas camadas, e pesos de conexões ajustáveis, comumente via o algoritmo de aprendizagem 'backpropagation', tem sido apontados pela literatura como um potente instrumento para classificação de padrões. Tal arquitetura, com n elementos na camada de entrada, e m elementos na camada de saída pode ser vista como uma aplicação unívoca

$$\varphi : \Re^n \rightarrow \Re^m \qquad (1)$$

A capacidade de aprendizagem de uma rede neural, em relação a um padrão, é o quanto a resposta da rede encontra-se próxima da saída desejada (um padrão pré-determinado). Dizer que uma rede foi capaz de aprender um determinado padrão p, na forma de um par $(x_p, d_p) \in \Re^n \times \Re^m$, significa que, dado um real $\varepsilon > 0$, pode-se tornar $|\varphi(x_p) - d_p| \leq \varepsilon$. Normalmente, não pode-se exigir do aprendizado que

$\varphi(x_p) = d_p$, porém pode-se arbitrar uma margem de imprecisão pequena o suficiente para propósitos práticos.

Alguns resultados já foram estabelecidos com relação à capacidade de aprendizagem de uma rede. Tem-se como estabelecido o seguinte [1]:

1. Se uma aplicação consiste de uma coleção finita de pontos, uma rede de 3 camadas (1 camada oculta) é capaz de aprende-la.
2. Se uma aplicação é continua e definida sobre um domínio compacto, uma rede de três camadas é capaz de aprende-la..
3. Sob condições muito gerais, todas as aplicações que podem ser aprendidas por uma rede neural, podem ser aprendidas por uma rede de 4 camadas (2 ocultas).

Estas condições apenas exigem que se tenha um número suficiente de elementos na camada oculta. Além disso, foi verificado que muitas funções que não seguem os critérios acima, podem também ser aprendidas por uma rede de 3 camadas. Em particular, descontinuidades podem ser teoricamente toleradas sob todas as condições prováveis de serem encontradas na prática.

Estas condições cobrem a maioria dos problemas práticos. A grande dificuldade consiste no treinamento para o estabelecimento dos pesos. Embora teoricamente qualquer par de padrões a ser reconhecido possa ser treinado , quando a rede tiver que abstrair muitas características simultâneamente, o aprendizado fica comprometido em termos de eficiência e velocidade. Para o reconhecimento de padrões na forma geral proposta, a rede deve ser invariante a translações, efeitos de escala e rotações no plano. Além disso, em muitos casos, existe a possibilidade do contorno do objeto focalizado não estar bem definido, de acordo com o enfoque da grade sobre a imagem. A rede deveria também ser invariante a esta possível característica.

A prática tem demonstrado poucos resultados otimistas em relação ao treinamento de todas estas características. Isto exige normalmente um conjunto de treinamento muito grande e assim a velocidade de convergência fica muito comprometida, com grandes possibilidades de cair em mínimos locais ou problemas de paralisia da rede.

Diversos pesquisadores desempenharam esforços no sentido de resolver esta questão, procurando adequar o tratamento do reconhecimento de padrões, aproveitando o lado positivo das capacidades destas redes e embutindo outras formas auxiliares de tratamento que possam amenizar o trabalho de abstrair tantas características simultâneamente. Pao [3], sugere a introdução de redes de ligações funcionais como uma forma alternativa equivalente ao trabalho desempenhado pelas camadas ocultas da rede: o de abstrair características de padrões. A mais compelente razão para a escolha deste modelo é a do conhecimento a priori de certas transformações funcionais em que os dados irão expor suas características salientes. Spirkovska e Reid [2] aperfeiçoaram uma arquitetura de redes neurais de ordem mais elevada (HONNs), ou redes pi-sigma, cujo pré-processamento consiste de somas de produtos de elementos de entrada. Restringindo a ordem dos produtos a três, obtiveram sucesso em classificar padrões binários com 100% de acerto, invariantes em translação e rotação e escala, em quadros de pixels de até 128x128.

Filtragem de Ruídos e Detecção de Agrupamentos

A segmentação da imagem, com filtragem de ruídos e detecção de agrupamentos, é

executada por uma rede neural cuja arquitetura consiste de camadas conectadas lateralmente, como mostra a figura 1.

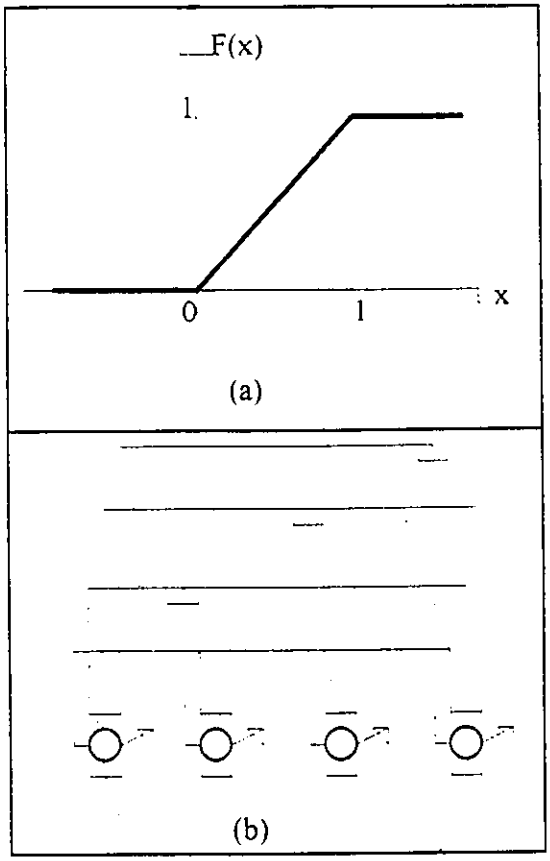


Figura 1. (a) Função de transferência do elemento de processamento. (b) Topologia da rede.

A dinâmica da rede consiste de processar o padrão de entrada, sendo que a saida de cada elemento de processamento alimenta as entradas dos demais elementos. Nesta camada, os elementos de processamento cooperam e inibem entre si os sinais de entrada, de acordo com o sinal algébrico dos pesos das conexões, iterativamente, até alcançarem um estado de estabilidade. A equação de iteração entre os elementos é:

$$x_i^{(k-1)} = F\left(x_i^{(k)} \div \frac{\sum_j x_j^{(k)} . w_{ij}}{\sum_j x_j^{(k)}} \right) \qquad (2)$$

onde:

k é o indice da iteração;

$x_i^{(k)}$ é o valor do sinal em processamento;

w_{ij} é o peso da conexão da saida do elemento i para a entrada do elemento j;

F é a função de transferência .

Para prevenir crescimentos de valores absolutos numéricos de forma desordenada e não-limitada, escolhem-se para F funções que mantem os limites superiores e inferiores de seus argumentos em um intervalo. Uma solução simples consiste em adotar a eqação (3).

$$F(x) = \begin{cases} 0 \text{ se } x < 0 \\ 1 \text{ se } x > 1 \\ x \text{ se } 0 \leq x \leq 1 \end{cases} \qquad (3)$$

Escolhendo adequadamente os pesos das conexões, a convergência de (3) fica assegurada para isto. Devem-se escolher pesos que refletem caracteristica de distribuição espacial e caracteristicas comuns a todos os padrões passiveis de serem classificados. Por exemplo: se os padrões são conexos no domínio, escolhem-se pesos positivos para as conexões de cada elemento com seu espacialmente vizinho e, eventualmente, para os próximos vizinhos. No entanto, pesos negativos de efeito inibitório devem encontrar-se presentes, para que exista efeito competitivo, pois caso contrário, o vetor converge com todos os valores para a unidade.

Desta forma, valores maiores tem a tendência de se fortalerecem ainda mais, e inibir outros eventuais agrupamentos a serem formados. De acordo com a força inibitória, um ou mais agrupamentos pode sobreviver. Estes são identificados como unidades dentro do vetor (que representa o padrão); isto permite isolar um agrupamento para análise individal. Ruídos, eventualmente fracos, deverão ser eliminados pelo processo competitivo.

Processamento do Agrupamento

A transformada de Fourier opera sobre valores continuos, e permite discretização preservando as propriedades fundamentais dos dados, o que torna viável seu uso computacional. Dado um vetor $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, a transformada de Fourier discreta de **x** resulta num vetor **X** cujas componentes, em igual número, podem ser calculada por (4), ou por métodos mais eficientes, como por exemplo a transformada rápida de Fourier [4]. Observe-se que o vetor da transformada discreta é complexo.

$$X_n = \frac{1}{N}\sum_{k=0}^{N-1} x_k e^{-j2\pi nk/N}, \quad (4)$$

para $0 \le n \le N-1$

onde

j é a unidade imaginária;
N é o tamanho do vetor;
x_k é o elemento do vetor de entrada
X_n é o elemento no vetor de entrada

A propriedade mais importante, neste caso, refere-se ao deslocamento. Se **x** tem **X** por transformada, então, para uma translação inteira h, o vetor $y_k = x_{k-h}$ tem por transformada a equação que segue:

$$Y_n = X_n . e^{-j2\pi nh/N} \quad (5)$$

Desta forma, $|\mathbf{Y}| = |\mathbf{X}|$, ou seja, os vetores possuem a mesma amplitude. Esta propriedade, se aproveitada, confere à rede híbrida o aspecto de "functional link" mencionada por [3].

Classificação do Padrão

A classificação é obtida a partir de uma rede neural, com uma arquitetura de três camadas e alimentação para frente, treinada segundo o algoritmo de aprendizagem Backpropagation[5]. Treina-se a rede de acordo com as amplitudes das transformadas de Fourier discretas do conjunto de padrões de possível classificação. Desta forma, a rede responde aos dados provenientes do pré-processamento do agrupamento. De acordo com a propriedade de invariância da amplitude da transformada, a rede neural não necessitará aprender a abstrair esta característica.

Exemplificando a dinâmica da rede neural híbrida

Fase 1. Parte-se de um conjunto de padrões bi-dimensionais, com valores entre 0 e 1, os quais representam três classes distintas (figura 2). Estes padrões são apresentados a primeira arquitetura de rede neural, a qual produz padrões de saída mostrados na figura 3.

0.90	0.80	0.20	0.12
0.80	0.19	0.07	0.06
0.42	0.12	0.11	0.03
0.13	0.10	0.09	0.05

0.07	0.05	0.10	0.09
0.11	0.80	0.20	0.19
0.09	0.87	0.18	0.15
0.95	0.20	0.09	0.07

0.88	0.10	0.09	0.17
0.78	0.02	0.21	0.12
0.89	0.07	0.05	0.28
0.99	0.10	0.21	0.08

Figura 2. Padrões que caracterizam três classes diferentes: Classe I. Classe II e Classe III. respectivamente.

1	1	0	0
1	1	0	0
0	0	0	0
0	0	0	0

0	0	0	0
0	1	0	0
1	1	0	0
1	1	0	0

1	0	0	0
1	0	0	0
1	0	0	0
1	0	0	0

Figura 3. Detecção de agrupamentos. conforme padrões da figura anterior.

Fase 2. A área de interesse de cada padrão é obtida através do produto entre do padrão de entrada e o padrão de saída da fase anterior, como mostra a figura 4. Estes novos padrões são agora apresentados a uma camada funcional, a qual produz novos padrões de saída (figura 5).

0.90	0.80	0	0
0.80	0.19	0	0
0	0	0	0
0	0	0	0

0	0	0	0
0	0.80	0	0
0.09	0.87	0	0
0.95	0.20	0	0

0.88	0	0	0
0.78	0	0	0
0.89	0	0	0
0.99	0	0	0

Figura 4. Focalização da área de interesse.

2.69	1.96	0.71	1.96
1.96	1.75	0.61	1.09
0.71	0.61	0.51	0.61
1.96	0.19	0.61	1.75

2.91	2.13	0.83	2.13
1.02	1.94	1.73	0.51
0.99	0.86	0.73	0.86
1.02	0.51	1.73	1.94

3.54	3.54	3.54	3.54
0.21	0.21	0.21	0.21
0.00	0.00	0.00	0.00
0.21	0.21	0.21	0.21

Figura 5. Amplitudes da Transformada de

Fase 3. As amplitudes das transformadas de Fourier de cada padrão são apresentadas a outra arquitetura de rede neural, a qual encarrega-se da classificação. Como o exemplo, tomamos como conjunto de treinamento os padrões mencionados nas figuras anteriores, e como conjunto de teste os padrões representados na figura 6.

(a)

0.15	0.09	0.07	0.10
0.06	0.11	0.05	0.21
0.09	0.12	0.17	0.83
0.08	0.06	0.78	0.95

0.09	0.12	0.07	0.98
0.07	0.10	0.92	0.92
0.21	0.07	0.87	0.11
0.03	0.06	0.05	0.09

0.03	0.12	0.09	0.79
0.02	0.11	0.12	0.92
0.23	0.12	0.07	0.89
0.05	0.15	0.13	0.77

(b)

0	0	0	0
0	0	0	0
0	0	1	1
0	0	1	1

0	0	0	1
0	0	1	1
0	0	1	0
0	0	0	0

0	0	0	1
0	0	0	1
0	0	0	1
0	0	0	1

(c)

2.73	2.01	0.83	2.01
1.99	1.78	0.68	1.12
0.73	0.62	0.49	0.62
1.99	1.12	0.68	1.78

2.86	2.08	0.72	2.08
1.01	2.12	2.02	0.78
0.84	0.89	0.94	0.89
1.01	0.78	2.02	2.12

3.37	3.37	3.37	3.37
0.18	0.18	0.18	0.18
0.01	0.01	0.01	0.01
0.18	0.18	0.18	0.18

Figura 6. (a) Padrão de entrada da rede neural hibrida (normalizados). (b) Agrupamentos obtidos. (c) Amplitudes para classificação.

O resultado da classificação é mostrado no quadro 1 a seguir:

Saída Desejada			Saída Obtida		
1.00	0.00	0.00	0.95	0.04	0.02
0.00	1.00	0.00	0.05	0.95	0.02
0.00	0.00	1.00	0.03	0.03	0.95

Quadro 1. Padrões de saida da rede neural de classificação, referentes as três classes distintas: [100] - Classe I, [010] - Classe II e [001] - Classe III.

Conclusões

As literatura atual sobre modelos conexionistas e a necessidade de encontrar uma maneira de reconhecer padrões complexos (uni e bi-dimensionais). levaram a construção de uma rede neural híbrida, ou seja uma rede neural composta de diversas arquiteturas. Com os testes verificou-se que a distribuição de tarefas entre as redes provoca melhorias no reconhecimento de padrões. Como mostra o exemplo, a rede é capaz de reconher padrões com 100% de acerto, independente da dimensão e posição espacial.

Referências

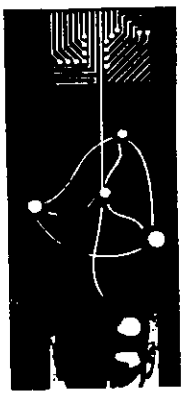
[1] MASTERS, Timothy. Practical Neural Networks Recipes in C++. San Diego, CA: Academic Press, 1993.

[2] SPIRKOVSKA, L. e REID, M. Connectivity Strategies for Higher-order Neural Networks Applied to Pattern Recognition. Proc Third Int'l Joint Conf. Neural Networks. San Diego, CA: June 17-21, 1990. I-21-I-26.

[3] PAO, Yoh-Han. Adaptative Pattern Recognition and Neural Networks. Reading: Addison-Wesley, 1989.

[4] BRIGHAM, E. Oran. The Fast Transform and its Applications. Englewood Cliffs, New Jersey: Prentice-Hall, 1988.

[5] RUMELHART, D. E., HINTON, G. E. e WILLIANS, R. J.. Learning Internal Representations by Error Propagation. In D.E. Rumelhart and J.L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstrutures of Cognition. Vol. 1: Foundations, pp.318-362, MIT Press, Cambridge, CA, 1986.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

LOW-OFFSET NEURAL WINNER-TAKE-ALL NETWORK

Volnei A. Pedroni

California Institute of Technology
Dept. of Electrical Engineering, 128-95
Pasadena, CA 91125 - USA
pedroni@romeo.caltech.edu

CEFET/PR
Depto. de Eletronica e Pos-Graduacao
em Informatica Industrial
Curitiba, PR - Brasil

Abstract - **Winner-take-all (wta) circuits are common building blocks in neural networks, vector quantizers, and other analog parallel signal processing systems. We present a wta circuit that employs a Hopfield-like architecture for the transmission of the positive-feedback coefficients over the 2-D computing array. The properties of this kind of network are further illustrated by means of a 32-input VLSI implementation on a 2.0 μm CMOS chip. Experimental results show a high voltage gain (so digital outputs are immediately available) and very small offsets (under 10mV in the worst-case scenario), with an analog dynamic range resolution of approximately 50 dB.**

I. INTRODUCTION

Analog hardware implementations of vector quantizers, content-addressable-memories, and other *n*-dimensional classifiers lead inevitably to system generation of an also *n*-dimensional set of electric signals which represent the results of some pre-defined vector distance metric computation. The remaining task is to identify which among these signals best satisfies the given metric, that is, to identify the greatest (or smallest) among the resulting signals. A variety of winner-take-all (wta) circuits, which perform this identification function, have been reported recently in the literature [1]-[3]. In this paper, we present a new wta network, which makes use of a neural architecture that closely resembles a Hopfield network [4] (Fig. 1) for the transmission of the positive-feedback coefficients over the 2-D computing array. The network operates in voltage-mode and is capable

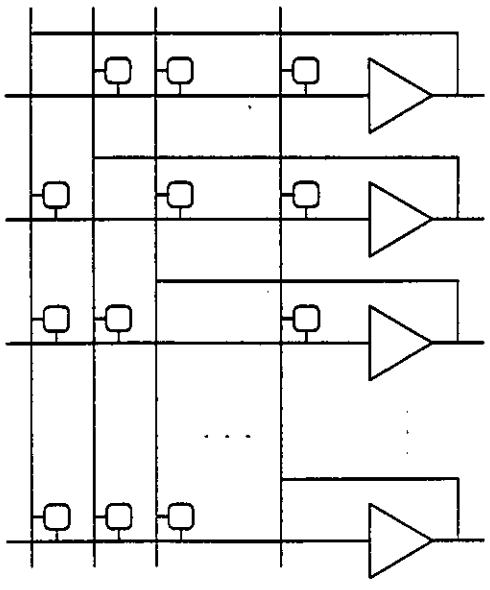


Fig. 1: Neural architecture adopted in the wta circuit.

of detecting very small input voltage differences, limited only by transistor mismatches, therefore overcoming the low detectability (low gain) of conventional $O(n)$ systems [1]-[2], and being also superior to other $O(n^2)$ implementations [3] in the respects that it does not require power supply switches and capacitors, presents higher gain and the inherent higher accuracy of voltage- over charge-mode systems, and also has the internal (computing) nodes completely insulated from the input terminals. The circuit is fully analog, so signals generated by binary processors like Hamming classifiers can be treated simply as a particular and more trivial case in which the input voltages are allowed to take on only certain values rather than any values. Although the interconnect complexity of this kind of circuit is $O(n^2)$, where n is the number of input signals (candidates), it requires just one transistor per synapse, therefore demanding very small silicon area for its implementation.

II. NEURAL WTA CIRCUIT

The neural wta circuit is shown in Fig. 2. It is composed of n amplifying cells (rows), each cell having n-1 active loads, cross-connected in an nx(n-1) array that closely resembles the network of Fig. 1. As can be seen, the circuit has analog inputs, $V_1, V_2, \ldots, V_n$, analog outputs $V_{O1}, V_{O2}, \ldots, V_{On}$, and digital outputs $D_{O1}, D_{O2}, \ldots, D_{On}$. The total current $nI_B = I_1 + I_2 + \ldots + I_n$ is set by the bias voltage V_B and is kept constant thanks to the common bus line V_C, thus providing a high voltage gain and high detectability for small perturbations; it also provides an output voltage which is independent of V_i (1). Preset transistors are also shown.

The principle of operation of the wta of Fig. 2 is based on a positive-feedback loop which only allows one winner in the equilibrium state. The basic operation of the network can be summarized as follows. If we suppose initially that $V_1 = V_2 = \ldots = V_n$, then $V_{O1} = V_{O2} = \ldots = V_{On}$ (theoretically only, due to transistor mismatches).

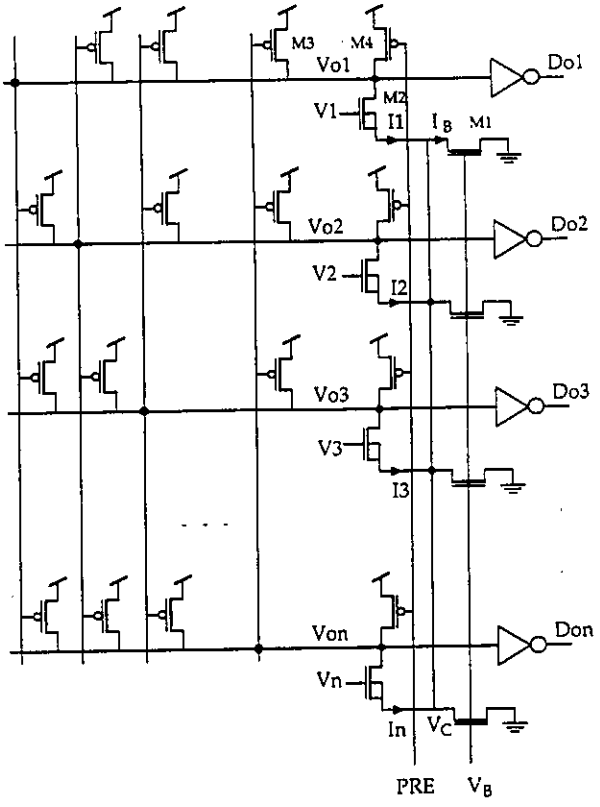


Fig. 2: Neural winner-take-all network.

If now we let one of the inputs, say V_1, increase, V_{O1} has to decrease in order to bring M2 of row 1 into linear mode, needed to keep I_1 constant ($V_{O2}, \ldots, V_{On}$ initially unchanged), so increasing the current driven by the leftmost P-transistor of rows 2 to n. However, the total current must remain the same, what forces the gate voltages of all the other P-transistors in rows 2 to n to increase, thus making the total current driven by the P-transistors in cell 1 smaller, what forces V_{O1} to become even lower.

Quantitatively, the behavior of the circuit can be summarized in the following way. Consider initially the state $V_1 = V_2 = \ldots = V_n \equiv V$ once again, in which case $V_{O1} = V_{O2} = \ldots = V_{On} \equiv V_{O=}$ (theoretically) and $V_C \equiv V_{C=}$. With all transistors in saturation and $\beta_j = (\mu C_{ox} W / L)_j$, we obtain that

$$V_{C=} = V_i - V_T - \sqrt{\lambda_1}(V_B - V_T) \quad \text{and}$$

$$V_{O=} = V_{DD} - V_T - \sqrt{\frac{\lambda_2}{n-1}}(V_B - V_T) \qquad (1)$$

where $\lambda_1 = \beta_1 / \beta_2$ (bias/input transistor) and $\lambda_2 = \beta_1 / \beta_3$ (bias/load transistor). From (1) we verify that, for the saturation condition to be satisfied for any input level, the transistor parameters must obey

$$\sqrt{\lambda_1} \le \frac{V_{i\min} - V_B}{V_B - V_T} \quad \text{and}$$

$$\sqrt{\frac{\lambda_2}{n-1}} \le \frac{V_{DD} - V_{i\max}}{V_B - V_T} \qquad (2)$$

If we let now one of the inputs (say V_1) grow, with all the others still alike (worst case), then V_{O1}, the winning output, decreases, while $V_{O2} = V_{O3} = \ldots = V_{On}$ increase. Call these two voltages levels V_L (low) and V_H (high), respectively. We know that, since M2 of row 1 and M3 of all the other rows are now in triode mode,

$$V_L \approx V_C \approx V_{C=} \quad \text{and} \quad V_H > V_{O=} \qquad (3)$$

A final consideration refers to the transistor parameter ratios λ_1, λ_2. If they are small enough, they may cause $V_H > V_{DD} - V_T$. If so, all of the P-transistors of row 1 (the winner) will tend to the cutoff state, in which case a preset mechanism (M4 in Fig. 2) is necessary in order to remove the circuit from this monostable state before executing the next computation. This situation exists if

$$\lambda_2 < \frac{(2V_{DD} - 2V_C - 3V_T)V_T}{(V_B - V_T)^2} \qquad (4)$$

which is independent from n, as expected, since in the monostable state each active cell is reduced to just one active load. The value of V_C in (4) can be obtained from (1), since $V_C \approx V_{C=}$.

III. EXPERIMENTAL RESULTS

A n=32 wta circuit was fabricated on a 2.0 μm MOSIS CMOS chip (Fig. 3). The transistors are all L=10 μm long, with $\lambda_1 = 0.5$ and $\lambda_2 = 5$. The experiments agreed consistently with the predictions, and the measured detectability was better than 10mV in the worst case, i.e., all inputs equal but one, with an input dynamic range slightly over 3V (50 dB).

A qualitative view of the experiments is shown in Fig. 4, with the upper trace of the scope showing the preset clock, the second channel showing the winning output, and the last channel showing one of the other outputs. As can be seen, the winning output becomes low when preset is released (PRE=5V), while the others stay high. The measurements show always only one winner present at a time, as imposed by the equilibrium state of the circuit.

The gain of the system is depicted in Fig. 5 by means of two separate measurements, which were performed with $V_B = 1.0\,V$ and $V_{REF} = 2.0V$, being V_{REF} the voltage applied to all the inputs except one, to which V_{in} was connected. The low system offset can be readily observed.

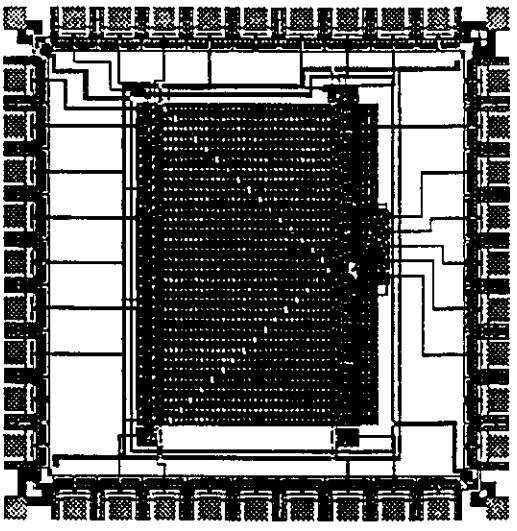


Fig. 3: n=32 wta on a 2.0μm CMOS chip.

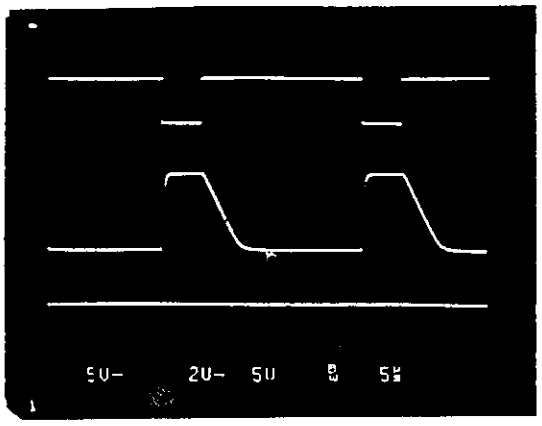


Fig. 4: Winning versus losing outputs.

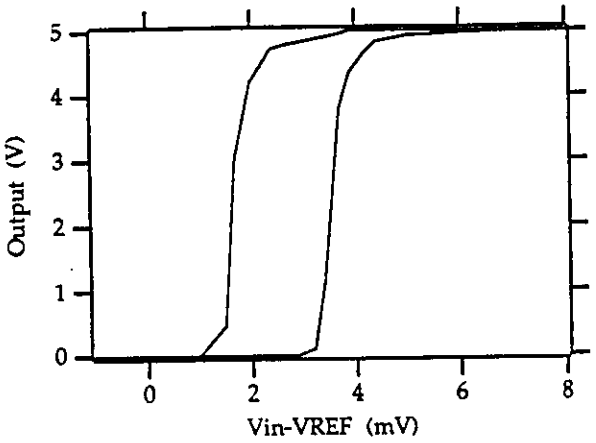


Fig. 5: Plots of two separate gain measurements.

V. CONCLUSION

We have made use of a neural architecture to introduce a new winner-take-all network. Basic properties of the circuit were qualitatively and quantitatively discussed and experimental results, obtained from a prototype chip fabricated using conventional 2.0 μm CMOS technology, were also presented. Positive-feedback, high gain, and small offset make possible the detection of very small perturbations, which are immediately decoded through digital outputs directly available on the network.

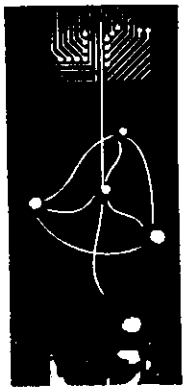
REFERENCES

[1] J. Lazzaro, S. Ryckebush, M. Mahowald, C. Mead, "Winner-take-all networks of O(N) complexity", NIPS, vol. 1, 1989.

[2] A. Andreou, K. Boahen, P. Pouliquen, A. Pavasonic, R. Jenkins, "Current-mode subthreshold MOS circuits for analog VLSI neural systems", IEEE Trans. on Neural Networks, vol. 2, 1991, pp. 205-213.

[3] Y. He, U. Cilingiroglu, E. Sanchez-Senencio, "A high-density and low-power charge-based Hamming network", IEEE Trans. on VLSI, 1993, pp. 56-62.

[4] J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", Proc. Nat. Ac. of Sciences, vol. 79, 1982, pp. 2554-2558.

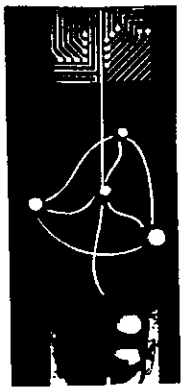


1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

PROCESSAMENTO DE IMAGENS

Anotações



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

Uma Solução Backpropagation Invariante a Escala, Rotação e Translação para o Reconhecimento de Caracteres

Luiz Eduardo Seabra Varella [1,2]
Emmanuel Piseces Lopes Passos [2]
Márcio Azevedo Santos [1]
Ricardo Lomba de Araujo [1]

[1] PETROBRÁS-Petróleo Brasileiro S.A.
Depex-Departamento de Exploração
Av. Chile 65, sala 1402
20035 Rio de Janeiro, RJ, Brasil
g073@c53000.petrobras.anrj.br

[2] IME-Instituto Militar de Engenharia
Seção de Sistemas e Computação/Cartografia
Praça General Tibúrcio, 80
22290 Rio de Janeiro, RJ, Brasil
pass@lncc.bitnet

Abstract: Character recognition represents one of the steps of major complexity in systems designed for the recognition of graphic documents (maps). The method suggested in this paper uses neural networks to recognize characters with variations of scale, rotation and translation. The method primarily consists of the extraction of invariant moments of image. These moments are subsequently submitted to a system of Backpropagation (multi-layer Perceptron and Backpropagation learning) networks. Such system consists of 466 networks, which classify the image characters. Results thus obtained reveal a high degree of accuracy to the extent that 97% of trained patterns were duly recognized.

1.0-Introdução

A disponibilidade de equipamentos matriciais de alta resolução (*Scanners*) e o crescente aumento da velocidade de processamento dos computadores, vêm estimulando nos últimos anos a investigação de procedimentos destinados ao reconhecimento de informações pictóricas. Tais procedimentos permitem que gráficos impressos em papel sejam convertidos em arquivos digitais codificados em formato vetorial.

A transferência de gráficos impressos para arquivos digitais, com objetivo de se alimentar Sistemas Gráficos, permite não só uma maior segurança dos dados, mas também facilita a reprodução, manipulação e análise das informações. Neste sentido, o presente trabalho propõe um procedimento automatizado que visa agilizar a aquisição de grandes volumes de dados espaciais.

O texto do trabalho foi organizado em três partes. A primeira, introduz o problema do reconhecimento de caracteres nos sistemas de digitalização de documentos gráficos. A segunda, apresenta uma solução, fundamentada em redes neurais, para o reconhecimento de caracteres com variações de escala, rotação e translação. Finalizando, a terceira parte descreve os resultados alcançados com a implementação.

2.0-Sistemas de Reconhecimento de Imagens

De maneira geral, os sistemas que realizam reconhecimento em imagens definem três etapas bem distintas: Pré-Processamento da imagem; Extração de *features* e Classificação da imagem. O Pré-Processamento se constitui da eliminação dos ruídos da imagem digitalizada, do afinamento (geração do esqueleto), da vetorização e da classificação dos elementos gráficos em duas categorias (linhas e caracteres ou símbolos). A Extração de *features* tem como objetivo mapear um espaço de medidas (imagem) em um outro, denominado espaço de *features*, de dimensões reduzidas. O objetivo principal da transformação não é apenas a redução da dimensionalidade do espaço de medidas, mas também, a descoberta de características que permitam a classificação da imagem, em particular os textos.

O sistema de reconhecimento de caracteres, aqui proposto, inicia-se com o cálculo dos seis primeiros momentos invariantes, mapeando-se os caracteres da imagem em um espaço de seis variáveis. Em seguida, estes momentos são aplicados em um sistema de redes neurais, fundamentado no paradigma *Backpropagation*, que responde a este estímulo de entrada com a classificação dos caracteres. As sub-seções seguintes descrevem, respectivamente, o extrator de característica utilizado e a arquitetura de redes neurais adotada para o reconhecimento de caracteres.

3.0-Momentos Invariantes

O uso dos momentos invariantes para extração de características fundamentais é uma técnica que remonta a alguns anos, sendo introduzida inicialmente por [Hu (1962)]. Sua utilização neste trabalho foi motivada pelas publicações de [Dudani (1977)] e [Khotanzad (1988)].

Conceitua-se por momentos invariantes o conjunto de funções não lineares invariantes à escala, translação e rotação, obtidas a partir dos momentos geométricos da imagem. Dada uma imagem digital g(x,y) de dimensões M x M, $\{g(x,y), x,y = 0,\ldots M\text{-}1\}$, o $(p+q)$-ésimo momento geométrico é dado por:

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} x^p y^q g(x,y) \quad \text{para } p,q = 0,1,2,\ldots .$$

Como são tratados caracteres de diferentes tamanhos, faz-se necessário a normalização do domínio de m_{pq}, que é feito mapeando-se a imagem plana M x M em um quadrado definido por $x \in [-1,+1]$ e $y \in [-1,+1]$. Conseqüentemente, a definição de m_{pq} fica:

$$m_{pq} = \sum_{x=-1}^{+1} \sum_{y=-1}^{+1} x^p y^q g(x,y).$$

Fazendo os momentos invariantes à translação, têm-se a seguinte formulação para os momentos centrais da imagem:

$$\mu_{pq} = \sum_{x=-1}^{+1} \sum_{y=-1}^{+1} (x-\bar{x})^p (y-\bar{y})^q g(x,y), \text{ onde}$$

$$\bar{x} = \frac{m_{10}}{m_{00}} \quad \text{e} \quad \bar{y} = \frac{m_{01}}{m_{00}}.$$

Normalizando os momentos centrais a fim de torná-los invariantes à escala chega-se a:

$$\eta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^{\gamma}}, \text{ onde } \gamma = \frac{p+q}{2} + 1.$$

O conjunto de seis funções não lineares, invariantes à escala, rotação e translação, é definido sobre η_{pq} da seguinte maneira:

$$\phi_1 = \eta_{20} + \eta_{02} ;$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4(\eta_{11})^2 ;$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 ;$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 ;$$

$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}) [(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) [3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] ;$$

$$\phi_6 = (\eta_{20} - \eta_{02}) [(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21}) .$$

Como os valores de ϕ_1 a ϕ_6 são extremamente pequenos, é comum, ao final do cálculo dos momentos, aplicar-se a função logarítmica a fim de se evitar problemas de precisão numérica. Logo, as características extraídas dos caracteres são definidas por: $\log |\phi_i|$, $i = 1,...,6$.

4.0-Descrição do Método de Reconhecimento de Caracteres

Antes de se descrever a solução, é de grande importância que se analise alguns aspectos. O primeiro deles refere-se à geometria e disposição dos padrões. Os textos de um documento gráfico apresentam-se de diversas formas, tamanhos e inclinações, portanto, deve-se chegar a uma solução que contemple tais características. O segundo diz respeito às espessuras. Vale lembrar que a imagem encontra-se representada por seu esqueleto, com todos os elementos apresentando espessuras unitárias. Esta particularidade deve ser levada em conta, uma vez que, a esqueletização da imagem produz sensíveis alterações na forma dos caracteres com inclinações não múltiplas de 45 graus. O terceiro, e último, refere-se à similaridade dos padrões. É importante que se desenvolva uma técnica que discrimine adequadamente caracteres semelhantes, por exemplo, I do 1, O do 0, B do 8.

Como em quase todos os sistemas neurais, baseados no modelo *Backpropagation*, faz-se necessário a realização de um conjunto de testes para se chegar a uma arquitetura definitiva. A título de simplificação para a implementação dos testes, foi digitalizado apenas um fonte de tamanho 0.4 cm, constituído de 36 caracteres. Para cada caracter foram geradas 13 imagens com inclinações múltiplas de 15 graus, variando de -90 à 90 graus, vide Figura 1. A seguir são descritos os testes realizados em um computador IBM/3090.

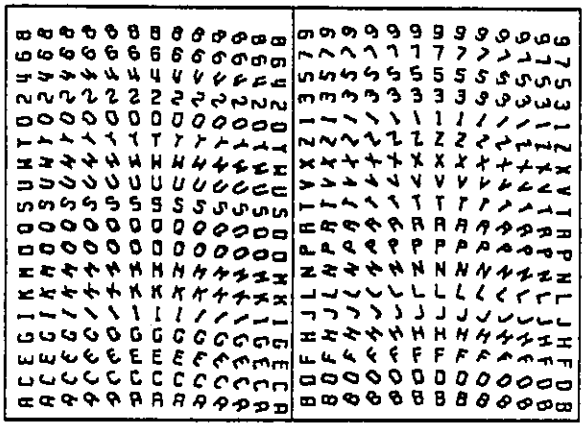


Figura 1 : Fonte Digitalizado

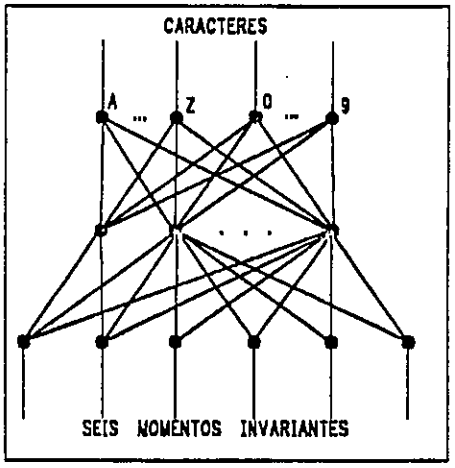


Figura 2: Teste 1 - Rede Única

4.1-Teste 1

O primeiro sistema que se pensou foi de apenas uma rede *Backpropagation* com seis elementos de processamento na camada de entrada, um elemento para cada momento invariante, e trinta e seis elementos na camada de saída, cada elemento respondendo por um caracter, vide Figura 2. Com intuito de se estimar um número inicial para as unidades de processamento da camada intermediária, foi treinado um conjunto de redes para o reconhecimento de caracteres com inclinação igual a 0 graus. Para este teste foram experimentadas 22 arquiteturas de redes, treinadas 20000 vezes com os padrões a 0 graus.

A tabela 1 mostra para cada número de elementos da camada intermediária o respectivo percentual de reconhecimento dos padrões

treinados. Foi adotada para a taxa de aprendizado, η, o valor 0.2 e para a constante *momentum*, α, o valor de 0.9. A realização deste teste consumiu aproximadamente 2h 10min de CPU.

De posse desse resultado, foram treinadas três redes com 250, 270 e 300 unidades de processamento na camada intermediária, visando o reconhecimento dos caracteres com todas as inclinações. Do conjunto de treinamento, 468 amostras, Figura 1, foram eliminados os dois W da extremidade (-90 e 90 graus) por serem idênticos aos padrões da letra M (90 e -90 graus). O treinamento destas redes consumiu mais de 32h de CPU, sem contudo apresentar resultados que justificassem o aprimoramento da solução. Este longo tempo de treinamento, sem dúvida, inviabiliza qualquer solução neste sentido.

Tabela 1 : Resultados do Teste 1.

Un. Inter.	% Rec.	Un. Inter.	% Rec.	Un. Inter.	% Rec.
10	2.5	108	61.5	216	71.7
16	2.5	120	66.6	230	82.0
32	10.2	130	79.4	250*	97.4*
40	7.6	140	71.7	260	87.1
56	25.6	150	74.3	280	84.6
68	30.7	170	79.4	300	79.4
80	51.28	186	71.79	-	-
92	48.71	200	82.05	-	-

4.2-Teste 2

Uma segunda alternativa foi a implementação de 36 redes, uma para cada caracter, com seis elementos de processamento na camada de entrada (seis momentos invariantes) e dois elementos na camada de saída, indicando a aceitação ou rejeição do padrão de entrada, vide Figura 3. Segundo esta arquitetura, o reconhecimento é feito aplicando-se os seis momentos invariantes do padrão em todas as redes. Conseqüentemente, a classificação é definida pela rede que apresentar o maior valor de ativação do elemento de saída responsável pela aceitação. Para o teste, as 36 redes foram treinadas com 466 amostras, sendo apresentadas 5000 vezes. A taxa de aprendizado (0.2) e o *momentum* (0.9) foram os mesmos do teste anterior. Para cada uma das redes foi pesquisado o melhor número de elementos da camada intermediária.

A tabela 2 mostra o resumo do teste, apresentando, para cada rede, o número de unidades escondidas e os percentuais *aproximados* de reconhecimento, indicados pelos escores de aceitação e rejeição dos padrões. Nesta tabela, os escores de aceitação e rejeição não expressam o resultado global do reconhecimento. Estes valores indicam que, por exemplo, para a rede que corresponde a letra A, 100% dos padrões A são reconhecidos e 99% dos demais padrões são ditos não A. Este índice de rejeição indica que existem alguns padrões não A que são classificados como A. Como apenas quatro redes apresentaram os percentuais ideais, ou seja, 100% para aceitação e 100% para rejeição, e algumas delas mostraram valores inferiores a 70%, é de se esperar que o resultado final aponte para a busca de uma outra solução. Apesar disso, 68% das amostras treinadas foram reconhecidas e o tempo de treinamento das 36 redes foi bem inferior (1h 25min) ao do teste de uma única rede, mostrando que é mais rápido treinar um conjunto de pequenas redes do que apenas uma com elevado número de elementos de processamento.

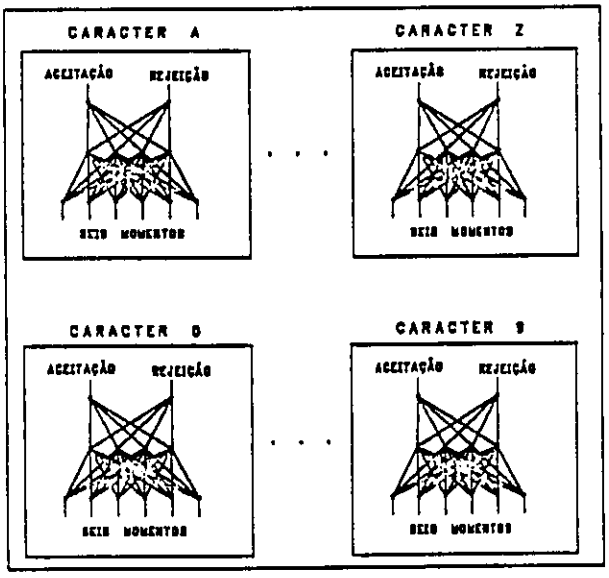


Figura 3: Teste 2 - 36 Redes.

Tabela 2 : Resultados do Teste 2.

Rede	% Ac.	% Rej.	Rede	% Ac.	% Rej
A-8	100	99	B-8	100	98
C-8	100	100	D-24	100	97
E-9	76	100	F-9	100	98
G-9	100	100	H-8	100	99
I-8	100	98	J-9	53	96
K-9	100	99	L-8	38	94
M-8	100	98	N-8	100	100
O-10	84	98	P-8	84	98
Q-8	84	95	R-8	100	100

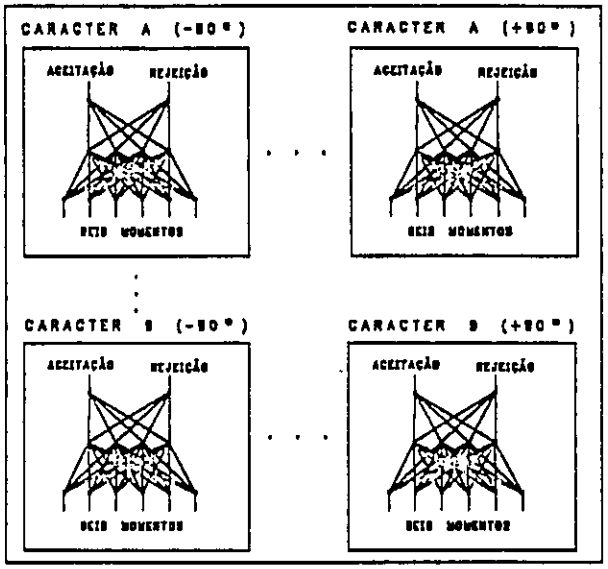


Figura 4: Teste 3 - 466 Redes.

4.3-Teste 3

Continuando nesta linha, com o propósito de se aumentar o percentual de reconhecimento, foi treinado um conjunto de 466 redes (36 caracteres * 13 inclinações de 15 em 15 graus - 2 W), com arquiteturas idênticas: seis elementos de processamento na camada de entrada, quatro elementos na camada intermediária e dois na camada de saída. Cada uma destas redes é responsável pelo reconhecimento de apenas uma inclinação de um dado caracter. A Figura 4 mostra o esquema de arquiteturas do sistema neural.

Quanto à estratégia de reconhecimento, o sistema segue o mesmo procedimento do teste anterior, ou seja, aplicam-se os momentos invariantes em todas as redes, e aquela que apresentar o maior valor de ativação no elemento de aceitação do padrão, classificará o caracter. O treinamento deste sistema consumiu 6h 25min de CPU. Para cada rede foram apresentadas 5000 vezes o conjunto de amostras, com a taxa de aprendizado igual a 0.2 e *momentum* igual a 0.9.

O resultado deste modelo, com 466 redes, mostrou avanços significativos, reconhecendo 97% dos padrões treinados. De maneira a se medir a capacidade de generalização do sistema, isto é, o reconhecimento de padrões não treinados, foram realizados outros testes descritos a seguir.

4.4-Teste de Generalização

Foi então gerada uma massa de padrões com caracteres rotacionados de 7.5 graus a partir dos padrões treinados. Para este conjunto de padrões, "não treinados", o sistema respondeu reconhecendo 41% dos caracteres. Este índice deveu-se ao elevado intervalo de rotação aplicado em cada caracter (15 graus) do conjunto de treinamento. De maneira análoga, foi realizado um segundo teste de generalização com intervalos de 10 graus para o treinamento e de 5 graus para o reconhecimento. A generalização deste modelo atingiu percentuais da ordem de 59%. Estima-se que com intervalos de 5 graus para o treinamento, a generalização atinja valores superiores a 80%. A tabela 3 sintetiza os resultados destes testes de reconhecimento dos padrões não treinados, mostrando para cada intervalo de rotação os respectivos percentuais de generalização.

Com relação ao reconhecimento de caracteres de diferentes tamanhos (escala), os percentuais se mantiveram no mesmo nível para variações até a metade do padrão treinado. Isto significa dizer que para o fonte utilizado nos testes, de tamanho igual a 0.4cm, pode-se reconhecer caracteres que variam de 0.2 a 0.6cm.

Tabela 3 : Teste de Generalização.

Int. Trein.	Int. Gener.	% de Rec.
15°	7.5°	41
10°	5°	59
5°	2.5°	> 80

5.0-Conclusões

Concluindo, o método aqui proposto para o reconhecimento de caracteres com variações de escala, rotação e translação se dá através de um conjunto de pequenas redes neurais, do tipo *Backpropagation*, onde cada uma delas é responsável pela classificação de uma única posição do caracter. Todas as redes são constituídas de seis elementos de processamento na camada de entrada, quatro elementos na camada intermediária e dois elementos na camada de saída.

É importante ressaltar que a acuracidade do reconhecimento é função da similaridade do conjunto de caracteres, da deformação produzida pela esqueletização, da resolução do dispositivo de aquisição da imagem e da capacidade de generalização introduzida ao sistema (quantidade de redes).

Referências

• L.E.S. Varella, Reconhecedor de Elementos Gráficos Digitalizados via Scanners, Dissertação de Mestrado, Instituto Militar de Engenharia, RJ, 1992.

• T. Pavlidis, Algorithms for Graphics and Image Processing, Computer Science Press, Rockville, 1982.

• K. Ramachandran, Coding Method for Vector Representation of Engeneering Drawings, Proc. of the IEEE, 1980, Vol.68, Num.7.

• D. Ting & B. Prasada, Digital Processing Techniques for Encoding of Graphics, Proc. of the IEEE, 1980, Vol.68, Num.7.

• L.A. Fletcher & R. Kasturi, A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images, IEEE Trans. on Patt. Anal. and Mach. Int., 1988, Vol.10, Num.6, PP.910-918.

• M. Hu, Visual Pattern Recognition by Moment Invariants, IRE Trans. Inform. Theory, 1962, Vol.IT-8, PP.179-187.

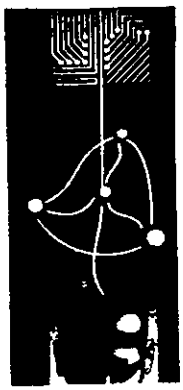
• S.A. Dudani "et alli", Aircraft Identification by Moment Invariants, IEEE Trans. on Computers, 1977, Vol.C-26, Num.1, PP.39-46.

• A. Khotanzad & J. Lu, Distortion Invariant Character Recognition by a Multi-Layer Perceptron and Backpropagation Learning, IEEE International Conference on Neural Networks, 1988, Vol.1, PP.625-632.

• P.K. Simpson, Artificial Neural Systems: Foundations, Paradigms, Applications, and Implementations, Pergamon Press, 1990.

• Y. Le Cun "et alli", Handwritten Digit Recognition Applications of Neural Chips and Automatic Learning, IEEE Commun. Mag., 1989, Vol. 27, Num. 11, PP. 41-46.

• D.E. Rumelhart, G.E. Hinton & R.J. (1986 b), Learning Internal Representations by Error Propagation, em D.E. Rumelhart & J.L. McClelland, Parallel Distributed Processing, 1988, Vol. 1, Cambridge, M.A., The MIT Press.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

Segmentação de Texturas Utilizando Operadores de Convolução e Redes Neurais

Evandro O. T. Salles*, Francisco J. N. Gomes**, Gutemberg H. Brasil***
*Departamento de Eng. Elétrica, **Departamento de Informática, ***Departamento de Estatística

Resumo: **Neste trabalho testou-se a eficácia das máscaras de Laws, juntamente com os operadores propostos por Pietikänien et al [1], para a segmentação de texturas. Estes operadores extraem características da imagem, que são utilizadas no treinamento de uma rede neural tipo *Counterpropagation*. Após a aprendizagem, uma imagem contendo várias texturas é varrida e seus pontos são classificados de acordo com a textura presente ao seu redor**

1. Definições sobre Texturas

A textura é uma característica bastante importante em imagens, podendo encontrar-se desde explorações multiespectrais de aviões e satélites até fotos microscópicas de cultivos celulares [2]. Utilizam-se as texturas para a análise de regiões homogêneas como fotos aéreas de zonas urbanas e zonas rurais, florestas, zonas desmatadas, etc. Também pode-se utilizá-las como um fator importante em segmentação de imagens ou na determinação de zonas defeituosas em superfícies de madeira [3].

Não existe um procedimento genérico para a interpretação de texturas. Para descrevê-las utilizam-se características tais como regularidade, granulosidade, repetitividade, aleatoriedade, linearidade, irregularidades, suavidade, rugosidade, etc. Estes são descritores qualitativos que podem representá-las. Porém, é difícil quantificar tais adjetivos utilizando-se um computador. Nem sempre nossas sensações visuais podem ser traduzidas de maneira matemática, simples e compacta. Além de todas as características e primitivas anteriores, não se deve esquecer da distribuição dos níveis de cinza sobre a textura. Ainda que seu conceito genérico a diferencie de sua cor, tal procedimento não é correto. Segundo Munõz [2], nível de cinza-textura são conceitos dependentes e se relacionam entre si de maneira muito parecida ao relacionamento onda - partícula. Do mesmo autor tem-se :

• Quando uma pequena área de uma imagem possui uma pequena variação das primitivas tonais, a propriedade predominante é o tom de cinza.

• Quando esta mesma área possui uma grande variação das primitivas tonais, a propriedade predominante é a textura.

Devemos adequar a ferramenta correta a um problema específico. Seja qual for o tipo de textura, existem técnicas diferenciadas para cada caso. Usa-se dividir sua análise em três categorias: análise estrutural, análise espectral e análise estatística [4]. Análise estrutural se refere ao estudo de primitivas e das relações topológicas entre elas. Este tipo de técnica se aplica a texturas altamente estruturadas e, sobretudo, com primitivas periódicas. Já a análise espectral utiliza as transformadas de Fourier para computar a frequência básica de toda a imagem, frequência de repetição de uma textura regular. As técnicas espectrais se baseiam na obtenção do espectro de Fourier e são usadas para identificar a periodicidade global de uma imagem, identificando-se picos estreitos e de alta energia que aparecem no espectro.

Ambas as técnicas se aplicam a texturas regulares que apresentam alguma lei de periodicidade de suas primitivas. Esta é uma característica bastante limitante. Na grande maioria dos casos, as imagens reais são misturas de texturas repetitivas e não repetitivas, sendo estas últimas de componente aleatório. Obviamente que as técnicas baseadas em medidas de regularidade não se adapam a estes tipos de texturas com facilidade. Ademais, em ambos os casos, realiza-se uma análise global sobre a imagem e, para tanto, deve-se supor que as frequências de repetição das primitivas não se modifiquem [5]. A pergunta que se faz é como garantí-lo ao longo de toda aimagem?

2. Máscaras de Detecção de Texturas

K. I. Laws, lançou uma metodologia baseada em máscaras de convolução padrão, sendo estas capazes de detectar segmentos horizontais e

verticais, pontos, etc. Este tipo de análise se comtrapõe à tradicional, baseada em pares de *pixels*, e se alinha com os micropadrões denominados *Textons*, propostos por B. Julesz em [6]. Para medir a energia de cada micropadrão, Laws propôs a utilização de estatísticas tipo ABSAVE, sobre uma região localizada (janela NxN), tendo como ponto central o *pixel* que carregará o resultado da convolução. Pietikäinen et al [1], estudando a eficácias das máscaras de Laws, propôs um novo conjunto de máscaras 5x5 baseadas não na definição quantitativa destas mas sim na sua forma geral. As novas máscaras são detectores simples de pequenos círculos concêntricos e de bordas centradas simetricamente em relação à máscara. Desta forma, conseguiu-se resultados melhores que apenas aplicando as máscaras de Laws 5x5. Os novos operadores, assim como os de Laws, possuem soma nula de seus componentes. No Apêndice podemos observar estes operadores

3. Procedimento

Para a aquisição de imagens empregou-se um *scanner*, já que não se dispunha de uma câmera e uma placa digitalizadora para barramento PC. O sistema foi desenhado para poder carregar uma imagem e permitir o treinamento da rede neural (*Counterpropagation*) com padrões significativos de texturas . Na fase de classificação, marca-se uma zona específica e sobre esta é realizada a segmentação de texturas.

Utilizou-se 64 tons de cinza. na representação da imagem. Esta não foi uma escolha e sim uma imposição do sistema computacional, já que não se encontrou um *driver* (para uma máquina tipo IBM-PC compatível) que pudesse manipular 256 níveis de cinza ao mesmo tempo. Cada imagem, contendo texturas, foi criada segundo os moldes mostrados na figura 1.

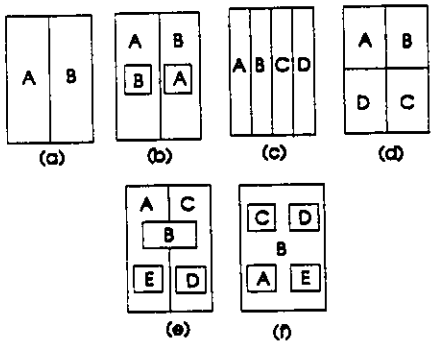


figura 1 - Moldes utilizados

Sobre as imagens são marcadas regiões retangulares (marcações em pontilhado) onde nestas se avalia a taxa de acertos obtida, através de tabelas. Utilizou-se uma rede neural *Counterpropagation* (CPN) do tipo *feed foward only* [7], [8], com mecanismo de consciência na camada de Kohonen [9]. O número de neurônios utilizado foi de, no mínimo, 3 vezes o número de classes a serem reconhecidas. A rede foi preparada para gerar uma cor na saída diferente, corespondendo a uma classe detectada. A rede CPN possui um rápido treinamento, se comparada com uma *Perceptron* Multicamadas + algoritmo de retropropagação. Por contar com uma camada de Kohonen, possui a característica de capturar a distribuição estatística dos dados de entrada.

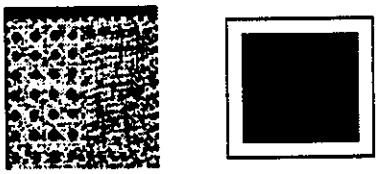
4. Testes e Resultados

Para os testes iniciais, utilizou-se as 9 (nove) máscaras propostas por Laws, de dimensões 3x3. Como estas possuem dimensões reduzidas, a computação de seus resultados é relativamente rápida. A configuração inicial do sistema (CPN + Laws + ABSAVE) segue na tabela abaixo:

Máscaras de Laws	3x3 - 9 máscaras
ABSAVE	9x9
n° de iterações	15000
Consciência	sim
Ptos. p/ Treino	80/(n° de classes)

Tabela 1 - Configuração inicial do sistema.

Na figura 2 observamos duas texturas, obtidas de [10]. Ambas são artificiais sendo que a textura da direita apresenta algumas distorções. Em 2(a) observamos as texturas a serem detectadas e em 2(b) o resultado da segmentação:



(a) (b)

figura 2 - Em (a) temos dois tipos de texturas onde podemos observar em (b) o resultado da segmentação. A CPN foi preparada para gerar uma cor na saída diferente para cada classe detectada

Na tabela 2 podemos observar as taxas de erros e acertos na classificação:

Resultado	Textura A	Textura B
Textura A	92.59%	7.41%
Textura B	6.39%	96.61%

tabela 2

Na figura 3 foram misturadas as texturas, para se analisar o poder de separação do sistema entre fronteiras. O molde usado refere-se ao apresentado na figura 1(b). A configuração é dada pela tabela 1. Como se observa, a grau de acerto foi elevado. Entretanto, existem alguns problemas nas fronteiras entre as texturas.

figura 3 - Texturas de papelão misturadas

Na a solução do problema proposto na figura 3, modificou-se o tamanho da janela de ABSAVE (de 9x9 passou-se a 7x7), para que o sistema pudesse ficar mais sensível às fronteiras. Com isto, perdeu-se um pouco a capacidade em detectar variações de uma mesma textura. Isto explica as pequenas regiões mais escuras presentes em regiões mais claras. Neste caso, pode-se melhorar a aprendizagem, aumentando-se a quantidade de pontos a serem treinados, ou mesmo selecionando-os melhor durante o treinamento. Existem 4 áreas marcadas para avaliação conforme mostra o molde desenhado na figura 4. A1, A2 e B1, B2 são as texturas A e B.

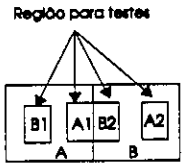


figura 4 - Molde das áreas de testes usado na figura 3

Para a região B1 os resultados foram:

Resultado	Textura A	Textura B1
Textura B1	10.74%	89.26%

Tabela 3 (a)

Para a região A1~B2:

Resultado	Textura A1	Textura B2
Textura A1	100%	0%
Textura B2	32.94%	67.06%

Tabela 3 (b)

Observe que na segunda linha da tabela 3 (b), obtivemos uma baixa taxa de acertos (67.06%), junto a uma alta taxa de erros (32.94%). Isto representa uma inadequação dos parâmetros do sistema ao tentar resolver fronteiras.

Na próxima imagem, figura 5, foram misturadas 4 tipos de texturas [10] segundo o molde apresentado na figura 1(c). Estas texturas apresentam poucas tonalidades de cinza.

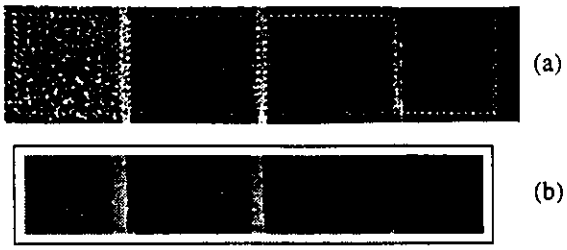


figura 5- Quatro tipos de texturas + fundo.

Ao tentarmos segmentar as texturas da figura 5, utilizando-se a configuração inicial, observou-se que à medida em que esta se tornava mais grossa, a janela de 9x9 perdia a capacidade em detectar o tamanho básico das microtexturas. Por isto, modificamos o tamanho original da janela de ABSAVE para 13x13 e obtivemos o resultado apresentado em 5(b). Na imagem original, podemos observar uma parte mais clara entre cada textura. Esta parte se refere ao fundo. Uma maneira de fazer com que o sistema separe as texturas do fundo é permintindo que este seja aprendido. Observe que apesar da separação ter sido boa entre as texturas A e B, o mesmo falha para C e D. Os resultados obtidos podem ser vistos na tabela 4

Res.	Text. A	Text. B	Text. C	Text. D
Text. A	94.95%	2.05%	1.74%	1.26%
Text. B	7.68%	85.58%	1.50%	5.24%
Text. C	12.54%	13.25%	74%	0.30%
Text. D	1.72%	0.87%	2.02%	95.39%

tabela 4

Na figura 6 observamos a presença de 5 (cinco) texturas mescladas em uma só imagem. As tres texturas menores (os retângulos menores) foram obtidas de tecidos. Na obtenção de cada uma delas, variou-se a resolução para podermos criar a sensação de textura grossa e textura fina. Novamente é possível notar problemas relativos com a resolução de fronteiras.

figura 6 - Exemplo onde se encontram 5 tipos diferentes de texturas

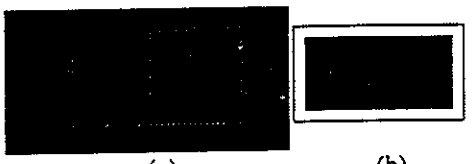
Como na fronteira, tanto as máscaras de Laws como a janela de ABSAVE estão entre uma e outra textura, a classificação penderá para aquele tipo que fornecer maior energia. Este problema aparece e aparecerá em todos os exemplos que fizermos. Para o retângulo superior e central, composto por uma

textura mais grossa, houve uma taxa maior de erros. Em especial, para esta área, obteve-se a tabela 5:

Res. (%)	Text. A	Text. B	Text. C	Text. D	Text. E
Text. B	22.63 (%)	65.39 (%)	7.33 (%)	2.38 (%)	2.27 (%)

tabela 5

Para o problema da figura 7 (molde figura 1(a)), existe um fator complicador que é a distribuição dos tons de cinza ao longo da imagem. Como na digitalização utilizamos uma resolução elevada, não capturamos toda a característica da distribuição dos níveis de cinza. Desta forma, a distribuição de manchas escuras e claras sobre a imagem se comporta como um processo estocástico não estacionário.



(a) (b)

figura 7 - Em (b) observamos o resultado da segmentação sobre a imagem propota em (a).

Neste caso, utilizou-se as 4 máscaras 5x5 propostas por Laws e as 8 máscaras 5x5 propostas por Pietikäinen et al [1]. O vetor de características passou a ter dimensão 13 (12 das máscaras + 1 para a normalização). A janela de ABSAVE empregada possuía dimensões 19x19, para 32000 passos de treinamento. O tempo de processamento (classificação) aumentou consideravelmente (o tempo de treinamento da rede continuou da ordem de 2 minutos, para 80 padrões de testes). Os resultados obtidos são apresentados na figura 7(b), com a taxa de acertos dada pela tabela 6.

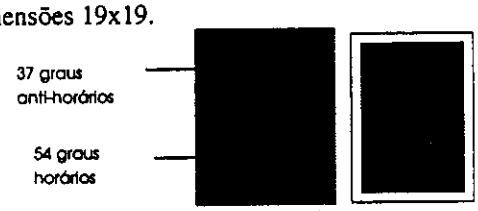
Dada a relação tom de cinza - textura, a propriedade que mais se evidenciou neste caso foi a textura. E isto era de se esperar pois a resolução com o qual a imagem foi obtida é elevada, fazendo com que as primitivas texturais se tornassem mais evidentes. Apesar das dimensões de ABSAVE, o sistema conseguiu detectar, com bons resultados, a fronteira presente entre as duas texturas.

Resultado	Textura A	Textura B
Textura A	95.61%	4.39%
Textura B	94.80%	5.20%

tabela 6

Um questionamento que pode ser feito neste momento se refere à característica das máscaras de Laws: sua detecção se baseia na trama ou em zonas com distribuição mais ou menos uniforme de tons de cinza? No conjunto de máscaras selecionado estão embutidos operadores que detectam ambas as propriedades, mesmo porque esta é a própria essência da análise e segmentação de texturas. No caso da segmentação das texturas presentes na figura 3 (tipos diferentes de papelão), observa-se que as primitivas são menos definidas, prevalecendo a distribuição tonal. Entretanto, tentar segmentar as duas texturas usando-se somente a comparação dos tons de cinza com um decisor (*Threshold*) pode levar a resultados muito piores que os apresentados. Ainda que a textura (ou a trama) não esteja claramente definida, a própria distribuição dos tons de cinza se encarrega de definir uma interrelação que, apesar de aparentemente aleatória, guarda em si alguns tipos de características marcantes. São estas as características detectáveis pelas máscaras empregadas.

Com o próximo teste observa-se a robustez do sistema quanto à rotação. Na figura 8, duas regiões entre texturas foram giradas de 37° anti-horários e 54° horários, conforme se observa. Ensinou-se o sistemas apenas com exemplos referentes à região não rotacionada. Como se observa, os resultados obtidos revelam que as características extraídas são robustas à rotação. Empregou-se as 12 máscaras de dimensões 5x5 (Laws + Pietikäinen) e janelas de ABSAVE de dimensões 19x19.



(a) (b)

figura 8 - Texturas rotacionadas

Como as máscaras detectam microcaracterísticas (pontos, pontos concêntricos, retas, etc, o que se assemelha ao proposto por [6]) e como não se mede interrelação entre *pixels*, obtemos medições absolutas (medimos a quantidade de energia de cada característica), o que favorece a robustez à rotação.

Para o caso da figura 9 [12], estamos utilizando o molde dado por 1(a).

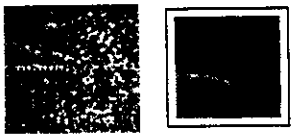


figura 9 - *Imagem contendo 4 tipos de texturas*

Observe que a rede falha sobretudo ao tentar classificar as texturas B e C. Ambas possuem uma predominância de tons de cinza muito parecidos, diferenciando-se principalmente na textura propriamente dita (trama). A distribuição de tons de cinza produz um efeito similar ao apresentado na

figura 7, onde tons mais escuros destoam do resto da imagem. Isto nos leva a concluir que para melhorar a segmentação, neste caso, é necessário ater-nos na trama. Assim, o sistema exige uma melhor sintonia, devendo-se testá-lo para um número maior de iterações (utilizou-se a configuração básica dada pela tabela 5.1, porém 12 máscaras de dimensões 5x5 - Laws + Pietikäinen), com uma maior quantidade de exemplos de treinamento para cada textura. A tabela 7 mostra a taxa de acertos obtida.

Res.	Text. A	Text. B	Text. C	Text. D
Text. A	88.3%	10.3%	1.33%	0%
Text. B	15.2%	62.53%	22.27%	2.52%
Text. C	19.2%	36.45%	44.11%	0.27%
Text. D	1.42%	3.01%	23.8%	71.7%

tabela 7

No problema a seguir, figura 10, imprimiu-se, usando-se instrumentos metálicos, tipos diferentes de texturas sobre uma superfície de madeira. O molde utilizado é dado pela figura 1(f). Como se obseva, a falta de definição nas bordas tornou-se bastante crítica (apesar das texturas não terem sido impressas segundo um molde retangular). O tamanho das máscaras de Laws e da janela de ABSAVE se mostram inadequadas (5x5 e 19x19 respectivamente) mas com elas se obteve os melhores resultados. Pode-se tentar aumentar a quantidade de iterações da CPN ou mesmo ajustá-la melhor. Neste caso, utilizou-se 15.000 ietrações.

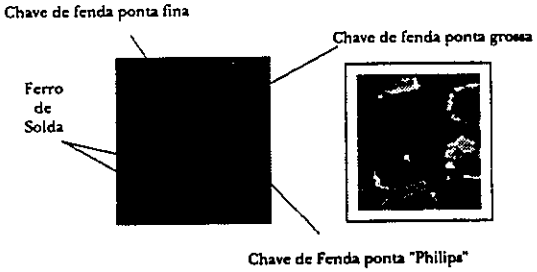


figura 10 - *Texturas impressas sobre madeira.*

Um outro problema se refere a pequenas regiões da textura de fundo que não foram detectadas como tal. A causa para isto pode ser devida a poucos exemplos de treinamento (foram usados 10) ou uma má seleção dos exemplos.

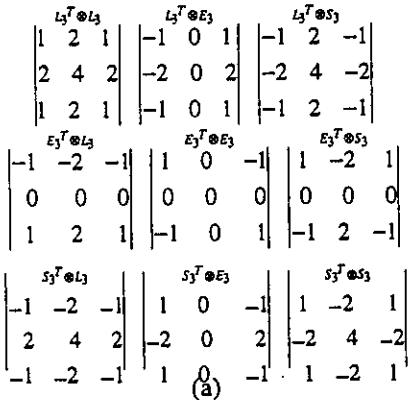
5. Conclusão

Uma escolha correta dos dados a serem usados no treinament é fundamental. Em nosso caso, sempre buscamos treinar a CPN com aqueles representantes de uma classe que mais se distanciavam de seu padrão. Com isto, a rede teve chance de encontrar uma superfície de separação que melhor se adaptava ao problema. Neste sentido, uma rede do tio LVQ pode apresentar resultados melhores.

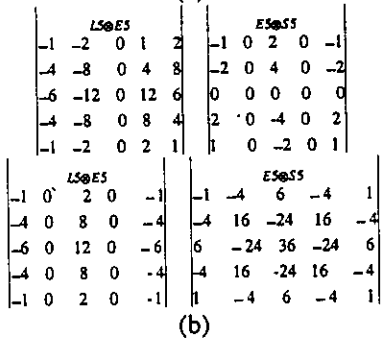
A configuração inicial da rede, dada pela tabela 1, apresentou bons resultados. Alguns testes foram feitos utilizando-se 5000 iterações, porém só funcionaram para problemas onde a segmentação era simples de ser obtida. Para o problema mostrado na figura 2, 1000 iterações mostrou-se suficiente, dando inclusive taxas de acertos comparáveis às apresentadas pela tabela 2 Entretanto, problemas como os que se observam na figura 7 exigiram 32000 iterações. Não necessariamente o número de padrões a serem treinados devem ser iguais para cada textura. Existem texturas que exigem uma quantidade maior de padrões para treinamento, devido a sua complexidade.

Hsiao et al [13] sugere que seja utilizado um procedimento de relaxação espacial para melhorar a detecção de texturas. Acreditamos que tal mecanismo possa ser empregado utilizando-se uma rede neural.

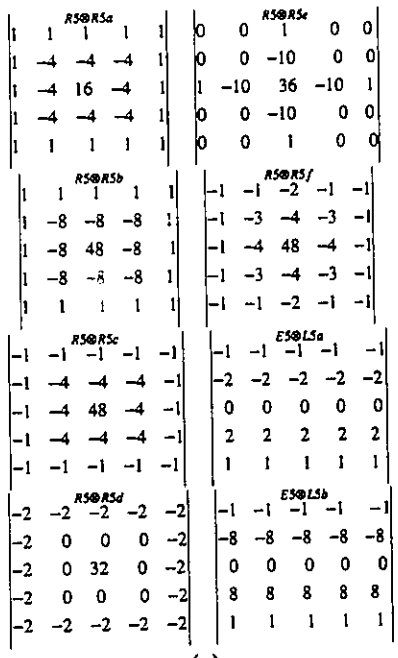
Apêndice



(a)



(b)



(c)

(a) Máscaras de Laws 3x3. (b) Máscaras de Laws 5x5. (c) Máscaras propostas por Pietikäinen et al [1]

Referências

[1] Experiments with Texture Classification Using Average of Local Pattern Matches
M. Pietikänien, A. Rosenfeld, L. S. Davis - IEEE Transactions Systems, Mans, and Cybernetics - vol SMC-13, May/June , n. 3, 1983, pag 421~426

[2] Jerarquización de Estructuras de Nivel Bajo y Medio para Reonociminto Visual. Aplicaciones a Texturas y Formas
J. A. Muñoz Blanco - Tesis Doctoral - Universidad Politécnica de Canarias - Facultad de Informática - Noviembre de 1987

[3] Identifying and Locating Surface Defects in Wood: Part of an Automated Lumber Processing System
R. W. Conners, C. W. McMillin, K. Lin, R. E. Vasquez-Espinosa - IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI -5, no. 6, November 1983

[4] Digital Image Processing
R. C. Gonzalez, Paul Wintz Addison-Wesley Publishing Company 1987

[5] Computer Perception of Repetitive Textures
L. G. C. Hamey - PhD Thesis - Carnigie Mellon, February 1988

[6] Textons, The Fundamental Elements in Preattentive Vision and Perception of Textures
B. Julesz, R. Bergen - Bell Sys. Tech. J., vol 62, no. 6, July/August 1983 pag 1619~1645

[7] Counterpropagation
R. Hecht-Nielsen - Aplied Optics, vol. 26, No. 23, December 1987

[8] Introduction to the Theory of Neural Computation
J, Herts, A. Krogh, R. G. Palmer - Addison-Wesley Publishing Company 1989

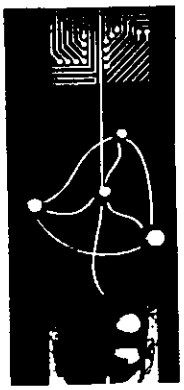
[9] Introdução à Computação Neuronal
L. P. Calôba - 9° CBA - UFES - minicurso - Setembro de 1992

[10] Segmentation of Textured Images and Gestalt Organization Using Spatial/Spatial-Frequency Representation
T. R. Reed, H. Wechsler - IEEE Pattern Analysis and Machine Intelligence - vol. 12 - no. 1 - January 1990, pag 1~12

[11] Local Linear Transforms for Textures Segmentation
M.l Unser - Signal Procesing , 11, (1986), pag 61 ~ 79

[12] Digital Image Processing
W. K. Pratt - A Willey Intercience Publication 1991 2nd edition

[13] Supervised Textured Image Segmentation Using Feature Smoothing and Probabilistic Relaxation Techiniques
J. Y. Hisao, A. A. Sawchuk - IEEE Transactions On Pattern Analysis And Machine Intelligence - Vol. II No. 12 December 1989 - pag 1279 ~1292



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá. 24 a 27 de outubro de 1994

Um Método de Aprendizado para Redes Neuronais Analógicas de Hopfield com aplicação à Compactação de Imagens

Jane Tavares Alvarez
ILTC,Instituto de Lógica,Filosofia eTeoria da Ciência,R.Almirante Teffé,637,
CEP 24030 080,RJ,Brasil
Luis Alfredo Vidal de Carvalho
COPPE,Universidade Federal do Rio de Janeiro,Caixa Postal 68511,RJ,Brasil

Abstract. Este trabalho apresenta e avalia três novos algoritmos de aprendizado para Redes Neuronais Analógicas de Hopfield [1][2], utilizando padrões gerados aleatoriamente. Os testes mostraram que, dada uma rede neuronal analógica de Hopfield com *n* neurônios, o número máximo de padrões estáveis pode superar os 15%*n* [1]. Em seguida, como uma aplicação do algoritmo de aprendizado de melhor desempenho, treinaram-se grupos de imagens de 400 pixeis nas cores preto, cinza e branco, para em seguida serem compactadas/descompactadas, via rede de Hopfied. Os testes nesta fase também produziram bons resultados: primeiro, todas as imagens foram 100% aprendidas; segundo, as distorções produzidas na etapa de compactação/descompactação, mesmo utilizando uma alta taxa de compactação, não inviabilizaram a compreensão das imagens.

Introdução. Em geral, o aprendizado em redes neuronais consiste apenas na obtenção de um padrão de conexão W que permita associar padrões eficientemente. Apresentaremos neste trabalho, três algoritmos de aprendizado para redes neuronais analógicas de Hopfield [4], dois dos quais não se restringem tão-somente a obter uma matriz de pesos satisfatória. Assim, buscando maior liberdade, que em outros tipos de rede pode significar apenas um aumento do número de conexões. trabalharemos também variando o potencial limiar θ e o ganho γ (dos estados de ativação dos neurônios da rede) a fim de que o aprendizdo seja mais eficiente.

Estes três algoritmos trabalham, respectivamente, variando W [3], W e θ , e por fim, W, θ e γ. No entanto, apenas este último algoritmo será abordado de forma mais completa, pois além de ser obtido a partir dos anteriores, ele demonstrou, através dos testes realizados, uma *performance* superior.

Intuitivamente, o que se propõe é encontrar uma matriz de pesos W, um potencial limiar θ e um ganho γ tal que dados *m* padrões de atividade $v_1, v_2, \ldots, v_m$ se consiga ajustar a função de energia E, definida pela equação [1], a todos estes pontos, de forma que eles constituam mínimos locais.

Consideremos uma rede neuronal analógica de Hopfield com *n* neurônios $N_1, N_2, \ldots, N_n$, *m* padrões de atividade, capacitância C, resistência R e impulso total de entrada u_i , relativo ao neurônio N_i , dado pela equação:

$$C\frac{du_i(t)}{dt} = \sum_j w_{i,j}\, a_j(t) - \frac{u_i(t)}{R},$$

sendo que $v_i \in (0,1)$ representa o estado de ativação do neurônio N_i e $W = [w_{i,j}]$

onde $w_{i,j} = w_{j,i}$, os pesos sinápticos, significam que o padrão de conexão W é uma matriz simétrica.

Assim, para uma matriz W, limiar θ, ganho γ e padrão de atividade $v \in (0,1)^n$ a rede possui energia

$$E(W,\theta,\gamma,v) = \frac{-1}{2}v^T W v + \frac{1}{R}\sum_{i=1}^{n}\int_{y=0}^{y=v_i} s^{-1}(y)dy$$

$$\text{onde } s^{-1}(y) = \frac{-1}{\gamma}\ln\left(\frac{1}{y}-1\right)+\theta. \qquad [1]$$

Diz-se que um padrão v é *mínimo local* de $E(W,\theta,\gamma,v)$, se e somente se,

1. $\nabla_v E(W,\theta,\gamma,v) = 0$
2. se a matriz hessiana[1] $H(W,\theta,\gamma,v)$ de $E(W,\theta,\gamma,v)$ com respeito a v é do tipo *semi-positiva*, isto é, se $x^T H(W,\theta,\gamma,v)x \geq 0$, $\forall\, x \in \Re^n$.

Em termos práticos, é difícil satisfazer a condição 2 para qualquer $x \in \Re^n$, principalmente pelo fato de $\Re^n$ ser incontável. Portanto, nos deteremos apenas a primeira condição, apesar de não ser possível assegurar a minimalidade da função de energia, isto é, que um dado padrão de atividade v é de fato, mínimo da equação:

$$E = \frac{-1}{2}\sum_{i}^{n}\sum_{i}^{n} w_{i,j}v_i v_j + \sum_{i}^{n}\frac{1}{R_i}\int_0^{v_i(t)} s^{-1}(v_i)dv_i$$

$$\text{onde } s^{-1}(v_i) = \frac{-1}{\gamma}\ln\left(\frac{1}{v_i}-1\right)+\theta_i. \qquad [2]$$

O problema então consiste em, dados m padrões de atividade $v^1,\ldots,v^m \in (0,1)^n$, determinar uma matriz de pesos W simétrica, um novo potencial limiar θ e um novo ganho γ, de forma que $v^1,\ldots,v^m$ representem pontos de mínimo, ou pontos próximos aos pontos de mínimo, da superfície definida pela função E de energia (equação 2).

Algumas Considerações Importantes. Seja $S^{-1}(v)$ um vetor cuja i-ésima componente é dada por v_i, onde $v_i \in (0,1)$. Levando em conta apenas a primeira condição da definição dada anteriormente, pode-se ter que se v é um padrão estável de $E(W,\theta,\gamma,v)$ então v é um padrão estável sobre uma matriz de conexões W, um potencial limiar θ e um ganho γ. Portanto, se v é um padrão estável de $E(W,\theta,\gamma,v)$ então $\nabla_v E(W,\theta,\gamma,v) = 0$, ou seja, $Wv - \frac{S^{-1}(v)}{R} = 0$. Mas isso ocorre, se e somente se,

$$\left(Wv - \frac{S^{-1}(v)}{R}\right)^T\left(Wv - \frac{S^{-1}(v)}{R}\right) = 0.$$

Logo, dados m padrões de atividade, se eles são todos simultaneamente estáveis então

$$\varepsilon = \sum_{p=1}^{m}\left(Wv^p - \frac{S^{-1}(v^p)}{R}\right)^T\left(Wv^p - \frac{S^{-1}(v^p)}{R}\right)s$$

endo $\varepsilon = 0$. Para facilitar, seja $\delta^p = Wv^p - \frac{S^{-1}(v^p)}{R}$. Então, da equação anterior podemos ter que

$$\varepsilon = \sum_{p=1}^{m}(\delta^p)^T\delta^p \quad \text{e} \quad \varepsilon = 0$$

Nosso problema agora se reduz a obter um ε mínimo a fim de que $v^1, v^2,\ldots, v^m$ estejam enfim, bem próximos de pontos estáveis. Dessa forma, é preciso minimizar $\varepsilon = f(W,\theta,\gamma,v^1,v^2,\ldots,v^m)$ com respeito à W, θ e γ, ou seja, obter $\min_W\varepsilon$, $\min_\theta\varepsilon$ e $\min_\gamma\varepsilon$.

Minimizar ε com respeito à W significa calcular $\nabla_W\varepsilon$, isto é, $\nabla_W f$. Dessa forma,

$$\nabla_W f = \sum_{p=1,m}\delta^p(v^p)^T + v^p(\delta^p)^T$$

Analogamente, $\min_\theta\varepsilon$ significa obter o valor de $\nabla_\theta\varepsilon$. Assim,

$$\nabla_\theta f = \sum_{p=1}^{m}\frac{-2}{R}\delta^p$$

E por fim, para minimizar ε com respeito a γ, basta calcular

$$\nabla_\gamma f = \sum_{p=1}^{m}\frac{-2}{R\gamma^2}\delta^p \ln\left(\frac{1-v^p}{v^p}\right), \qquad \gamma \in \Re_+^n$$

[1] $\nabla_v E$ significa o gradiente de E com respeito a v. A matriz hessiana é a matriz das derivadas segundas.

Os Algoritmos. À medida que se aumentavam os graus de liberdade da rede neuronal, um novo algoritmo surgia. O primeiro algoritmo criado foi obtido variando-se o padrão de conexão W. Utilizando-se o potencial limiar, deu-se origem ao aprendizado em W e θ. Finalmente, variando W, θ e γ gerou-se o algoritmo de aprendizado em W, θ e γ.

Algoritmo de Aprendizado em W, θ e γ

$W \leftarrow W_{inicial}$
$\theta \leftarrow \theta_{inicial}$
$\gamma \leftarrow \gamma_{inicial}$
enquanto f puder ser localmente minimizada **faça**
$W \leftarrow W - \eta_W \nabla_W f$, para algum $\eta_W > 0$
$\theta \leftarrow \theta - \eta_\theta \nabla_\theta f$, para algum $\eta_\theta > 0$
$\gamma \leftarrow \gamma - \eta_\gamma \nabla_\gamma f$, para algum $\eta_\gamma > 0$
fim_do_enquanto

Uma vez que buscamos os pontos de mínimo da função f, iremos considerar $-\nabla f$. Assim, W, θ e γ variam proporcional e respectivamente, às taxas de aprendizado η_W, η_θ e η_γ, conforme $-\nabla f$.

Os critérios de parada. A condição de parada dos algoritmos consiste em ter o $\nabla_* f$ nulo para * = W, θ e γ, conforme o algoritmo considerado. Mas, $\nabla_* f = 0$ implica em $\|\nabla_* f\| = 0$. Como em termos computacionais não é fácil garantir que $\nabla_* f = 0$, utiliza-se um ERRO* considerado aceitável em relação aos mínimos procurados, e que possibilitou um tempo razoável de processamento para os algoritmos. Dessa forma então, o terceiro algoritmo pára quando $\frac{\|\nabla_W f\|}{m} < ERRO_W$, $\frac{\|\nabla_\theta f\|}{m} < ERRO_\theta$ e $\frac{\|\nabla_\gamma f\|}{m} < ERRO_\gamma$, isto é, quando $\|\nabla_* f\|$ atinge valores bem próximos de zero.

Resultados Experimentais. Segundo Hopfield [1], dada uma rede neuronal com n neurônios, o número máximo p de padrões estáveis que se pode obter é $p \approx 0.15n$. Mostraremos aqui, no entanto que, com os algoritmos propostos é possível, fazer com que este valor seja superado de maneira significativa.

Cada algoritmo de aprendizado foi testado utilizando-se 50 neurônios e m padrões de atividade gerados aleatoriamente. Para cada algoritmo foi feito um levantamento em função do número de padrões simultaneamente aprendidos. Assim, dados m padrões calcula-se uma fração ou porcentagem conforme o número m^* de padrões que tenham sido simultaneamente aprendidos.

Concluído o aprendizado, ocorre a fase de simulação em que a rede evolui dinamicamente até que o valor da energia computacional, definida pela equação [2] atinja um mínimo satisfatório, onde enfim estabiliza. No entanto, a fim de que a simulação tivesse um caráter ainda mais preciso, resolvemos considerar a norma quadrática média da diferença entre dois padrões de atividade em tempos consecutivos, ou seja,

$$\frac{\|V_{t+\Delta t} - V_t\|^2}{n}$$

Mesmo assim, poderia haver o risco de que a simulação fosse processada em um tempo muito aquém do necessário. Dessa forma, considerou-se que esta etapa deveria levar um tempo de simulação mínimo igual a 0.1, o que equivale a 100 iterações.

Para o primeiro algoritmo, foram considerados nulos a matriz W inicial e o potencial limiar θ, enquanto que R = C = γ = 1.0. Segundo os resultados dos testes que foram realizados para até m = 15 padrões, temos que, a partir de m = 3 começa a haver uma interferência gradual entre os padrões a serem ensinados. No entanto, o desempenho do algoritmo supera os 15% dos 50 neurônios utilizados, pois ao serem fornecidos à rede m = 10 padrões é possível ensinar todos os 10 em 50% dos casos, além de ser possível, no pior caso, ensinar $m^* = 8$ em 20% dos casos. Mesmo ao serem dados m = 15 padrões, é possível

ensinar, simultaneamente, $m^* = 13$ padrões em 50% dos casos e $m^* = 11$ padrões em 20% dos casos, apesar de não serem simultaneamente aprendidos todos os 15 padrões.

Os valores da taxa de aprendizado η_W são completamente empíricos, variando à medida que se aumenta a quantidade m de padrões a serem treinados e se deseja manter um bom nível de treinamento.

No segundo algoritmo, tanto o padrão de conexão W inicial quanto o potencial limiar inicial foram gerados aleatoriamente. No entanto, γ, R e C assumiram o mesmo valor constante igual a 1.0, como no algoritmo anterior.

Os testes foram realizados para até m = 25 padrões e apresentaram um desempenho superior ao do algoritmo 1, o que se atribui essencialmente, a um maior grau de liberdade no aprendizado. Por exemplo, dados m = 9 padrões, foram simultaneamente aprendidos todos os 9 em 60% dos casos, enquanto que pelo primeiro algoritmo isso ocorreu apenas em 40% dos casos. Já ao se considerar m = 15 padrões é possível aprender todos os 15 em 30% dos casos e $m^* = 9$ padrões em 10% dos casos. Considerando m = 20 ou m = 25, foi possível ensinar, respectivamente, $m^* = 20$ e $m^* = 25$ em 20% dos casos testados.

Para o último dos algoritmos a ser testado, utilizou-se $W_{inicial}$ e $\theta_{inicial}$ aleatórios, γ_i inicial igual a 0.5, para $1 \leq i \leq 50$, sendo preservados os valores de R = C = 1.0. A variação de γ permitiu um aprendizado ainda melhor em relação ao obtido pelos demais algoritmos. Por exemplo, dados m = 15 padrões foi possível ensinar todos os 15 padrões em 50% dos casos contra 0% e 30% obtidos respectivamente, pelos algoritmos 1 e 2. Ainda, ao serem utilizados m = 20 padrões foi possível ensiná-los todos em 30% dos casos, sendo também possível (no pior caso) ensinar $m^* = 15$ padrões em 20% dos testes realizados.

Pela análise dos testes realizados, concluímos empiricamente que, o algoritmo de aprendizado em W, θ e γ apresenta um desempenho superior em relação aos demais. Dessa forma, iniciaram-se os testes de aplicação, que consistiram em utilizar grupos de uma até nove imagens de 400 pixeis, ensiná-las, depois compactá-las e finalmente, restaurá-las. Entretanto, é importante notar que, cada pixel de uma imagem qualquer *I* representa na verdade, o estado de ativação de um neurônio que a constitui.

No que diz respeito ao aprendizado destas imagens, que serão melhor detalhadas adiante, temos que não houve o mínimo possível de degradação, isto é, o aprendizado foi 100%, inclusive nos casos em que se utilizou um número maior de imagens para serem simultaneamente aprendidas.

O Método de Compactação Utilizado. O método de compactação aqui utilizado é bastante intuitivo e está intimamente ligado ao uso da rede neuronal, apesar de existirem outros métodos [5] que não necessitam da rede para realizar tal tarefa.

Suponhamos uma imagem *I* constituída por n pixeis ou neurônios. Considerando uma taxa de armazenamento de neurônios t_a, sabemos que dos *n* pixeis ou neurônios que formam a imagem, apenas $t_a * n$ neurônios serão armazenados. Este método propõe, então, que sejam recrutadas as unidades que adquiriram maior conhecimento a respeito de *I*.

A fase de descompactação de I consiste em fornecer à rede a imagem I_c compactada, que constitui parte da informação total. Como a propriedade fundamental de uma rede neuronal de Hopfield é a memória de acesso pelo conteúdo (CAM), basta fornecermos à rede a imagem compactada I_c, para que ela evolua até atingir um padrão estável, ou seja, uma imagem I_f descompactada.

Cabe agora compararmos a imagem final I_f com a imagem I_o previamente concebida no início de todo o processo. Tal comparação é feita analisando-se unidade a unidade, isto é, o pixel ou neurônio N_i^o de I_o e o pixel ou neurônio N_i^f de I_f, e assim

sucessivamente. Quando as unidades N_i^o e N_i^f, para $1 \leq i \leq n$, são distintas, computamos o valor do número de neurônios errados (NNE), que é uma forma adequada e prática se considerarmos o aspecto visual da imagem.

Os Testes de Compactação. As imagens originais I_o foram criadas com 400 pixeis na cores preto, cinza e branco. Visto que cada pixel constitui na verdade um neurônio N_i com um estado de ativação $a_i(t)$ pertencente ao intervalo (0,1), foi preciso relacionar cada cor da imagem I_o com um valor de ativação neste intervalo. Assim, o primeiro passo foi dividir o intervalo (0,1) em três sub-intervalos, atribuindo a cada cor um valor no meio do subintervalo.

Na representação das imagens aprendida (gerada ao final do treinamento e da simulação) e descompactada, o processo foi inverso, uma vez que os estados de ativação deviam ser mapeados em uma das cores usadas. Neste caso então, a cor foi fornecida conforme o subintervalo do intervalo (0,1) em que se encontrava os estados de ativação, já que cada sub-intervalo podia ser relacionado com uma das cores usadas.

A compactação foi realizada para grupos de até nove imagens previamente aprendidas pela rede neuronal, utilizando-se para as taxas de compactação valores que variaram entre 0.9 e 0.1, conforme a necessidade em armazenar, respectivamente, um número menor ou maior de neurônios.

Foi possível observar que, à medida que aumentamos o número de imagens a serem compactadas, a taxa média de compactação em que o número de neurônios (pixeis) errados atinge zero, diminui progressivamente. Isso ocorre porque, durante o processo de aprendizado a rede neuronal obteve conhecimento acerca de todas as imagens simultaneamente, não guardando necessariamente, as características específicas de cada uma delas. Além disso, ao se dar o processo de compactação são guardados os neurônios mais fortes que compõem cada imagem, podendo em muitos casos, os estados de ativação relativos aos neurônios mais marcantes de uma imagem estarem bem próximos das ativações dos neurônios mais significativos de outras imagens. Portanto, quanto maior o número de imagens no processo de compactação, maior a quantidade necessária de pixeis a serem armazenados, a fim de que haja o mínimo possível de superposição de imagens e consequentemente, um bom nível na qualidade das imagens restauradas.

Considerando a relação entre a taxa de compactação e a média do número de neurônios errados, dado um número k de imagens que foram simultaneamente apendidas, onde $2 \leq k \leq 9$, temos que, à medida que diminuimos a taxa de compactação, a média do número de neurônios ou pixeis errados (MNNE) também diminui, até atingir zero. No caso de $k = 1$, a MNNE foi nula logo ao considerarmos 90% de compactação. Ao se aumentar o número k de imagens a serem compactadas e computando a média do número de neurônios errados, temos que esta média tende a baixar mais lentamente à medida que incrementamos o número de imagens. Portanto, são necessárias taxas cada vez menores, conforme se queira utilizar um número maior de imagens, e assim obter uma MNNE bem baixa.

Conclusão. Os testes realizados com os três algoritmos de aprendizado propostos para redes neuronais analógicas de Hopfield, mostraram de forma empírica, que mesmo o mais simples desses algoritmos pode superar os resultados existentes até então. Dessa forma, o terceiro algoritmo, que obteve melhor desempenho entre os algoritmos propostos pôde, dados uma rede com 50 neurônios e 15 padrões de atividade gerados aleatoriamente, ensiná-los todos em 50% dos casos, sendo que no pior caso, foi possível ensinar 11 padrões dos 15 fornecidos.

A etapa de aplicação, embora tenha utilizado um número baixo de cores,

também produziu bons resultados, principalmente se considerarmos o aspecto visual das imagens.

Apesar de termos tratado com imagens pequenas de apenas 400 pixeis, por limitação de tempo e de máquina, podemos encarar estas imagens, em termos de trabalhos futuros, como sub-imagens formando imagens maiores que as utilizadas neste trabalho.

Bibliografia.

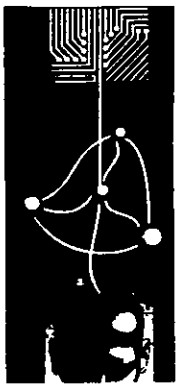
[1] Hopfield, J.J.,"Neural Networks and Physucal Systems with Emergent Collective Computational Abilities", Proc. National Academy of Sciences USA 79, 2554-2558., 1982.

[2] Hopfield, J.J., "Neurons with Graded Response have Collective Computational Properties like those of Two-state Neurons", Proc. National Academy of Sciences USA, 81, 3088-3092, 1984.

[3] Barbosa, V. C., Carvalho, L.A.V., "Learning in Analog Hopfield Networks", International Joint Conference on Neural Networks - IJCNN, 1991.

[4] Alvarez, Jane Tavares, "Um Método de Aprendizado para Redes Neuronais Analógicas de Hopfield com Aplicação à Compactação de Imagens", Tese de Mestrado, Rio de Janeiro, COPPE/UFRJ, 1992.

[5] Soares, Luis Fernando Gomes, et al., "Fundfamentos de Sistemas Multimídia", Porto Alegre: Instituto de Informática da UFRGS, VIII Escola de Computação, 1992.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

STRUCTURING NETWORKS FOR IMAGE CLASSIFICATION USING COMPETITIVE LEARNING

P. R. Green[†], C. L. Nascimento Jr.[‡] and T. A. York[†]
[†] UMIST, UK and [‡] Instituto Tecnológico de Aeronáutica, Brazil
E-mail: p.r.green@umist.ac.uk, cairo@ieav.cta.br, tay@umist.ac.uk

Abstract: One major problem facing the designer of a multilayer feedforward neural network for image classification is to determine the optimum number, and size, of the hidden layers. In this article we propose to add competition within the hidden layer to encourage structuring of the solution during Back-Propagation training. Simulation of an image classification network with added competition shows that a subset of hidden layer units account for most of the network solution. As a result, the remaining hidden layer units can be removed without causing significant performance degradation.

1. INTRODUCTION

One major problem facing the designer of a multilayer feedforward neural network for image classification is to determine the optimum number, and size, of the hidden layers. A popular approach is to initially design with one hidden layer, guess the number of units and use the Back-Propagation algorithm to train the network through adaptation of the weights, Hinton (1).

The Back-Propagation algorithm minimises the average squared output error over the set of training patterns without regard to the structuring of the solution in the network. Consequently, given an overestimated number of hidden layer units, Back-Propagation will distribute the solution over the whole hidden layer, contrasting with a structured solution which uses only the required minimum of units. The unstructured solution, requiring all hidden units, has clear associated implementation penalties and the potential for degraded classification of noisy images.

In this article we propose to add competition within the hidden layer to encourage structuring of the solution during Back-Propagation training. Simulation of an image classification network with added competition shows that a subset of hidden layer units account for most of the network solution. As a result, the remaining hidden layer units can be removed without causing significant performance degradation.

2. BACKGROUND

Multilayer feedforward neural networks have been successfully applied to a diverse range of data classification and recognition problems. Of these, image classification is a current focus as few 'traditional' computing solutions offer the robustness the problem demands, Zurada (2). In these networks, the hidden layer adapts to form a set of feature detectors that are then used by the output layer to classify the image. The adaptation of the hidden layer in developing these feature detectors is one of the major innovative components of the neural network approach.

In contrast to feedforward networks trained with supervised learning, such as the Back-Propagation algorithm, other neural network models use competition amongst the units of a layer and unsupervised learning to discover a set of feature detectors, Rumelhart and Zipser (3). The mutual inhibition between units of a layer in competitive learning encourages the partitioning of the input pattern space into features for which a minimum number of units are active. This unsupervised

classification, also termed *clustering*, is the basis of a number of neural models such as the Kohonen network, Kohonen (4), and the Adaptive Resonance Theory 1 (ART1) network, Carpenter and Grossberg (5).

The addition of competition transforms a feedforward network to a feedback network. Back-Propagation was originally conceived for feedforward networks (1), and has subsequently been extended to certain classes of feedback networks, Rumelhart et al (6) and Almeida (7). However, the network model presented in this report retains the Back-Propagation algorithm for feedforward networks.

3. BACK-PROPAGATION WITH ADDED HIDDEN LAYER COMPETITION

The proposed network model, shown in figure 1, consists of input, hidden and output layers, as found in the standard multilayer feedforward neural network, but differs in that the units of the hidden layer are also linked by competitive weights. Figure 1 shows the weights for a typical unit in each layer (I_q, H_p, O_m). The bias weights for the hidden and output layer units are omitted for clarity.

Feedforward Behaviour

Given that: NI, NH and NO are the number of input, hidden and output units respectively, $\boldsymbol{I}$ is the external input vector applied to the network and $\boldsymbol{H}$ and $\boldsymbol{O}$ are the outputs of the hidden and output units, then the vector $\boldsymbol{O}$ is calculated as follows:

$$\boldsymbol{I} = \left[I_1, \ldots, I_q, \ldots, I_{NI} \right]^T$$

$$\boldsymbol{H}^k = \left[H_1^k, \ldots, H_p^k, \ldots, H_{NH}^k \right]^T$$

$$\boldsymbol{O} = \left[O_1, \ldots, O_m, \ldots, O_{NO} \right]^T$$

$$\boldsymbol{net}_H = \boldsymbol{W}_{HI}\boldsymbol{I} + \boldsymbol{bias}_H \qquad (1)$$

$$\boldsymbol{H}^k = Sig\left(\boldsymbol{W}_{HH}\boldsymbol{H}^{k-1} + \boldsymbol{net}_H \right) \qquad (2)$$

$$\boldsymbol{O} = Sig\left(\boldsymbol{W}_{OH}\boldsymbol{H}^{NT} + \boldsymbol{bias}_O \right) \qquad (3)$$

where in the recursive eq. 2, k = 1, ..., NT, $\boldsymbol{H}^0$ is a zero vector, $Sig(\boldsymbol{X})$ is the sigmoid function, $[1+\exp(-x)]^{-1}$, applied to each component of the vector $\boldsymbol{X}$, and the input vector $\boldsymbol{I}$ is kept constant during the recursion. The weight matrices $\boldsymbol{W}_{HI}$ and $\boldsymbol{W}_{OH}$ have dimension NH by NI and NO by NH respectively. The column bias vectors $\boldsymbol{bias}_H$ and $\boldsymbol{bias}_O$ have dimensions NH and NO. The competition matrix $\boldsymbol{W}_{HH}$, dimension NH by NH, is given by:

$$W_{HH}^{ij} = \begin{cases} \gamma < 0 & if\ i \neq j \\ 0 & if\ i = j \end{cases} \qquad (4)$$

and γ is called the *competition factor*. Note that if the competition factor is set to zero, the network model reduces to the standard feedforward multilayer network.

Network Training

Given a set of NPAT training images and their desired classifications (T_{pat}, pat = 1, ..., NPAT), the network is trained using standard Back-Propagation. A training epoch consists of the presentation of all NPAT training images. For each training image I_{pat} presented, the outputs of the hidden layer H_{pat}^{NT} and the output layer O_{pat} are calculated according to eqs. 1, 2 and 3. The error signal, defined as the difference between the desired and actual network outputs for a given image, is then back-propagated through the network.

Training adapts the matrices $\boldsymbol{W}_{HI}$ and $\boldsymbol{W}_{OH}$ and the vectors $\boldsymbol{bias}_H$ and $\boldsymbol{bias}_O$ following the presentation of each image. The error signal is not allowed to back-propagate through the competitive weights in the hidden layer. The matrix $\boldsymbol{W}_{HH}$ is not adapted.

Following each epoch, we can form a measure of the classification error by calculating the Root-Mean-Square (RMS) Error over the set of NPAT training images as:

$$RMS\ Error = \frac{1}{NO}\sqrt{\sum_{pat=1}^{NPAT} E_{pat}^2} \qquad (5)$$

where:

$$E_{pat}^2 = \left(T_{pat} - O_{pat} \right)^T \left(T_{pat} - O_{pat} \right) \qquad (6)$$

4. AN IMAGE CLASSIFICATION PROBLEM

A series of simulations were designed around a simple image classification problem for the purpose of demonstrating the learning dynamics and knowledge structuring of the proposed network model.

The problem required sixteen unique bitmapped images to be classified using a 1-of-16 line coding scheme. The images used were taken from the 8 by 8 pixel character bitmaps used by the IBM PC XT, IBM (8). These were edited to remove the redundant last column and row to form the 7 by 7 bitmaps shown in figure 2. The sixteen characters images were drawn from the uppercase hexadecimal

character set (0,1-9,A-F).

The simulated network had an input layer with 49 units, a single hidden layer with 10 units and an output layer with 16 units. Each input unit was connected to a single pixel of the image bitmap with active pixels having an input value of 1 and inactive pixels 0. The output layer implemented the 1-of-16 line classification coding.

5. SIMULATIONS

The simulations consider two particular cases:

I. No competition, $\gamma = 0$
II. With competition, $\gamma = -1.4$

The network weights and biases, excluding the competitive weights, were initialized as small random numbers with a uniform distribution around 0. Excluding the competitive weights, in both cases the network initial weights and biases are exactly the same. The learning rate and momentum rate were set to 0.5 and 0.1 respectively. For the case with competition, the number of recursions through the hidden layer (NT) was set to 2.

Figure 3 shows the learning curve for the cases with and without competition for 1200 epochs. The RMS Error was calculated every 4 epochs. During each epoch, all 16 input patterns were presented with a different random ordering and without noise. The ordering of the presentations of the patterns is the same for both simulations.

In both cases the network learned to classify the input images. Figure 4 shows, for the case without competition, the output of the hidden layer at the end of training for all 16 training input patterns. Figures 5 and 6 show, for the case with competition ($\gamma = -1.4$), the output of the hidden layer following the first recursion (H^1) and following the second recursion (H^2).

6. DISCUSSION

Figure 4 shows that, for the case without competition, each hidden unit is activated by several images. Figure 5 shows that, when the competition was added, after the first recursion through the hidden layer (k=1), some hidden units, in particular units 2 and 10, have small outputs for all training inputs. Figure 6 shows that after the second recursion (k=2), hidden units 1,2,3,7 and 10 have zero output values.

Considering figure 6, it could be easily assumed that hidden units 1,2,3,7 and 10 could be removed from the network without any degradation in classification. This is not the case because the network is not simply feedforward. A near zero output from a hidden unit at the end of recursion does not imply near zero output on earlier iterations. Consequently, such a unit could have played an active role in the competition mechanism and can not therefore be removed.

From eq. 2, we can see that, due to the competition within the hidden layer and the fact that all hidden units use a sigmoid function, during recursion the output of a particular hidden unit will always be smaller or equal to its value after the first recursion cycle, ie, $H_i^k \leq H_i^1$.

From the above discussion, it can be seen that if the output of a particular hidden unit is zero after the first recursion cycle for all input training images then, such a hidden unit can be removed without any degradation in classification. Hence, the removal of a hidden unit will result in a small degradation if it has small output for all input training images after the first recursion.

This suggests that hidden units can be progressively removed in exchange for a measured degradation in classification. One possible criterion for selecting hidden units would involve a simple threshold function. Defining $H_i^{1\,MAX}$ as the maximum output, over all input patterns, from hidden unit i (i = 1, ..., NH) after the first recursion. Hidden unit i would be removed if:

$$H_i^{1\,MAX} < \theta \qquad (7)$$

where θ is a threshold level selected by the designer.

Tables 1 and 2 show the effect of selecting different threshold levels for the simulated networks trained with and without competition. The *relative RMS error* is defined as the ratio between the RMS error when a particular set of hidden unit (HU) is removed and the RMS error with no HUs removed. We define a classification as valid when a single unit in the output layer has an output: 1) greater than half of the summation of all output layer units and 2) greater than 0.5.

For the case without competition, H_i^{MAX} is near 1 for all hidden units. This indicates the need for large values of threshold and also that no hidden units can be removed without a significant performance.

TABLE 1 - Threshold for case without competition
(RMS Error no HUs removed = 0.0134)

Threshold θ	HUs Removed	Relative RMS Error	Misclass-ifications
0.9000	none	1.0	0
0.9971	3	4.7	1
0.9980	1,3	8.5	4
0.9981	1,3,5	10.7	8
0.9983	1,3,5,7	15.8	13
0.9990	1,3,5,7,9	19.4	14

TABLE 2 - Threshold for case with competition
(RMS Error no HUs removed = 0.0526)

Threshold θ	HUs Removed	Relative RMS Error	Misclass-ifications
0.10	none	1.00	0
0.40	2	1.02	0
0.60	2,10	1.17	0
0.70	1,2,10	1.55	2
0.89	1,2,7,10	2.21	2
0.90	1,2,3,7,10	2.72	3

TABLE 3 - Single HU failure without competition

Hidden Unit	RMS Error	Relative RMS Error	Misclass-ifications
-	0.013	1.0	0
1	0.050	3.7	0
2	0.116	8.7	4
3	0.063	4.7	1
4	0.105	7.9	5
5	0.071	5.3	2
6	0.112	8.3	5
7	0.079	5.9	1
8	0.100	7.4	2
9	0.070	5.2	2
10	0.116	8.6	4

TABLE 4 - Single HU failure with competition

Hidden Unit	RMS Error	Relative RMS Error	Misclass-ifications
-	0.053	1.0	0
1	0.058	1.1	0
2	0.054	1.0	0
3	0.063	1.2	0
4	0.106	2.0	2
5	0.216	4.2	7
6	0.219	4.2	8
7	0.064	1.2	0
8	0.213	4.0	8
9	0.213	4.0	7
10	0.056	1.1	0

degradation, as shown by the relative RMS error column.

In contrast, in the case with competition, there are units with relatively small H_i^{MAX} which can be removed without significant performance degradation.

One incorrect inference of the results given in table 2 would be that, training with competition has increased the tolerance of the network to the loss of *any* small group of hidden units. Consider the results of tables 3 and 4 where the effects of removing a single hidden unit are shown for networks trained with and without competition. The relative RMS error column for the case with competition reveals two distinct classes of hidden unit. Hidden units 1,2,3,7 and 10 form one class, where each could be removed with less than a 20% increase in the RMS error. Removal of any one unit in the other class, consisting of units 5,6,8 and 9, would cause a significant loss of classification performance.

The existence of distinct classes clearly shows that training with competition encourages structuring of the solution and not an increased tolerance of the network to the loss of *any* hidden layer unit.

For the network trained without competition, table 4 shows no particular divisions in the relative RMS error of the hidden units and hence no structuring.

CONCLUSIONS AND FUTURE WORK

This article has presented a neural network model which uses competition in the hidden layer of a feedforward network to encourage structuring during Back-Propagation supervised learning. The result of this structuring is to reduce, over the feedforward network, the number of hidden layer units active in feature detection. Compact structuring and fault-tolerance are seen to be conflicting requirements. These issues were demonstrated with the simulation of a simple image classification problem. The benefits offered by this new network model must be weighed against the need to tune an additional parameter, the competition factor. Simulations results not presented here indicate that too larger competition can inhibit learning.

There are a number of aspects of the new model that merit further work, including; modification of the learning algorithm to remove 'inactive' hidden layers units during training, scheduling the competition factor γ to improve the time taken to train the network to a given accuracy of classification, investigation of the effects of the number of cycles through the hidden layer (NT) on training and classification performance.

ACKNOWLEDGMENTS

The authors would like to acknowledge Dr. M. Zarrop for his interest in this work.

Cairo L. Nascimento Jr. gratefully acknowledges the support of the Brazilian Research Council (CNPq) under grant 200.617/88.5.

REFERENCES

1. Hinton, G.E., 1989, Artificial Intelligence, 40, 185-234

2. Zurada, J.M., 1992, "Introduction to Artificial Neural Systems", West Publishing Co.

3. Rumelhart, D.E. and Zipser, D., *Feature Discovery by Competitive Learning*. In Rumelhart, D.E. and McClelland, J.L. (eds.), 1986, "Parallel Distributed Processing", Vol. I, MIT Press, 151-193

4. Kohonen, T., March 1988, IEEE Computer Magazine, 11-22

5. Carpenter, G. and Grossberg, S., 1986, "Eighth Annual Conference of the Cognitive Science Society", 45-62

6. Rumelhart, D.E., Hinton, G.E. and Williams, R.J., *Learning Internal Representations by Error Propagation*. In Rumelhart, D.E. and McClelland, J.L. (eds.), 1986, "Parallel Distributed Processing", Vol. I, MIT Press, 318-362

7. Almeida, L.B., *Backpropagation in Non-Feedforward Networks*. In Aleksander, I., 1989, "Neural Computing Architectures", North Oxford Academic, 74-91

8. IBM, 1986, "Technical Reference: Personal Computer XT and Portable Personal Computer"

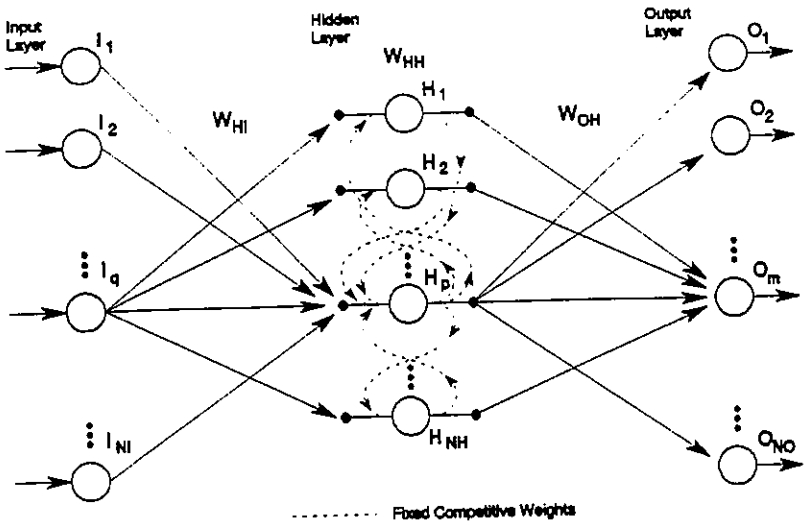


FIGURE 1 - Proposed Neural Network Model

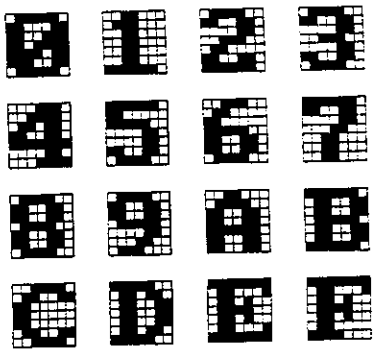


FIGURE 2 - Image bitmaps

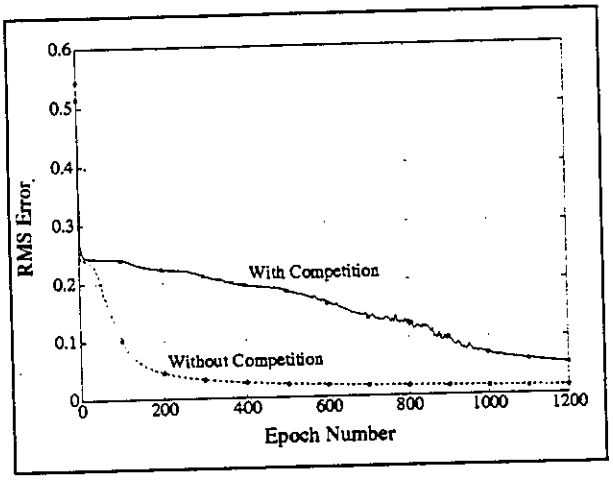


FIGURE 3 - Learning Curves

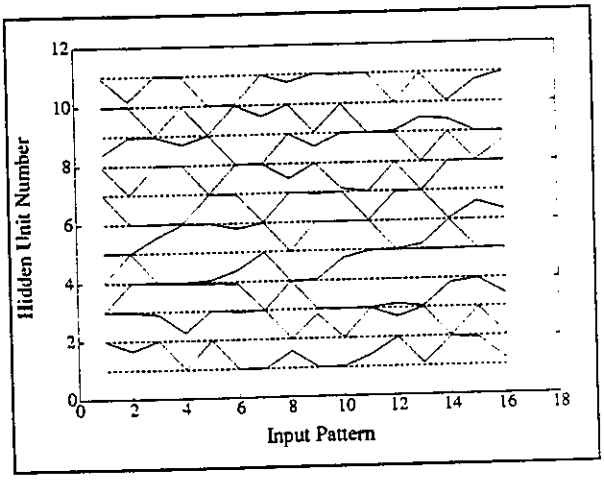


FIGURE 4 - Hidden Unit Outputs - No Competition

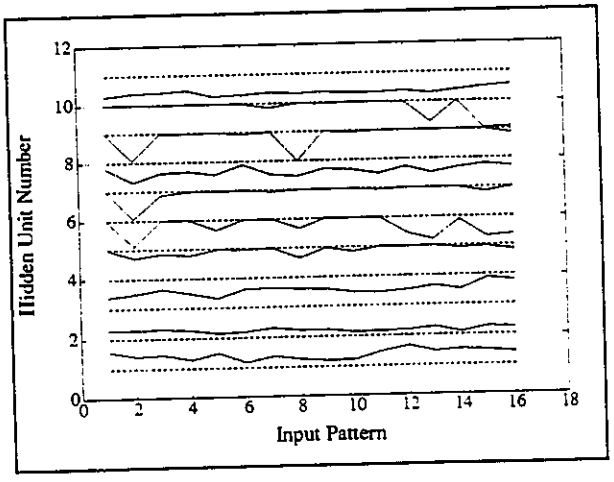


FIGURE 5 - Hidden Unit Outputs - With Competition First Recursion

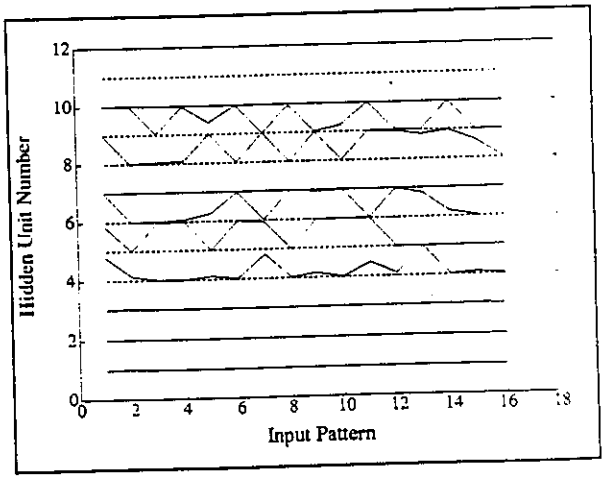
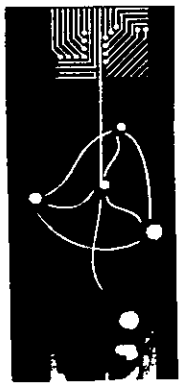


FIGURE 6 - Hidden Unit Outputs - With Competition Second Recursion

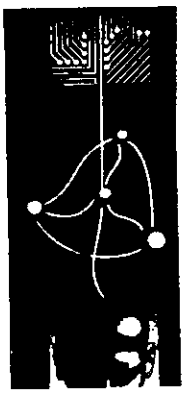


1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

VISÃO COMPUTACIONAL

Anotações



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajuba. 24 a 27 de outubro de 1994

Extração do Mapa de Direções de Impressões Digitais Via Rede Neural

FLÁVIO A. P. SOARES, RUI SEARA,
ORLANDO J. TOBIAS & JOSÉ C .M. BERMUDEZ

Laboratório de Instrumentação Eletrônica - LINSE
Departamento de Engenharia Elétrica
Universidade Federal de Santa Catarina
C.P. 476 - 88.040-900 - Florianópolis - SC - Brasil

RESUMO

Neste artigo, apresentamos uma nova metodologia para a extração do mapa de direções de impressões digitais, utilizando uma rede neural. Nesta estrutura, a imagem da impressão digital é segmentada em blocos e de cadà bloco extrai-se um conjunto de parâmetros que são apresentados à entrada da rede neural. Esta tem como resposta a direção preferencial encontrada para o referido bloco. O desempenho da rede obtida, em condições reais de operação, foi muito bom. Esta estrutura foi desenvolvida para ser utilizada em um Sistema Automático de Classificação de Impressões Digitais.

I. INTRODUÇÃO

O campo de pesquisa em processamento digital de imagens tem crescido consideravelmente nos últimos anos. Este avanço tem possibilitado resolver eficientemente inúmeros problemas nesta área. Neste mesmo campo, a aplicação de técnicas baseadas em redes neurais [1-3] vem contribuindo muito para que tarefas como inspeção visual e classificação de objetos possam ser automatizadas. Dentro deste escopo, está se desenvolvendo um Sistema Automático de Classificação de Impressões Digitais. Este sistema tem como objetivo automatizar as tarefas de aquisição, armazenagem e classificação de impressões digitais, hoje ainda realizadas totalmente de forma manual no Brasil. Inicialmente, prevê-se a digitalização das fichas dactiloscópicas existentes nos arquivos policiais, seguido de um processamento para extração do mapa de direções das principais linhas, obtendo-se assim, o esqueleto da impressão digital. Numa segunda etapa, a aquisição das impressões digitais será feita diretamente através de "scanners" óticos, especialmente desenvolvidos para este fim.

A classificação das impressões digitais é realizada através da observação de certos pontos característicos presentes neste esqueleto. Para se realizar as etapas de extração do mapa de direções e de classificação das impressões digitais, optou-se por uma abordagem via rede neural. Esta escolha foi baseada na capacidade de reconhecer padrões inerentes às estruturas neurais. Neste artigo é apresentada e avaliada a estrutura da rede neural desenvolvida para a extração do mapa de direções utilizado no nosso Sistema Automático de Classificação de Impressões Digitais.

II. PARÂMETROS DE ENTRADA

No processo de aquisição, as impressões digitais são capturadas através de um "scanner", obtendo-se uma imagem de 512 x 512 *pixels* com 256 níveis de cinza, como a ilustrada na Fig. 1. Esta imagem deve então ser "binarizada" (utilização de apenas 2 níveis de cinza),

promovendo-se, assim, uma redução importante do volume de dados no armazenamento e no tratamento das etapas seguintes do sistema. Para tal, foi utilizado um algoritmo especialmente desenvolvido para esta tarefa, que efetua a "binarização" da imagem, como também, a correção de pequenos defeitos oriundos do processo de adquisição.

A segunda etapa do processo é a extração do mapa de direções da imagem "binarizada". Para tanto, algumas técnicas propostas na literatura foram avaliadas. A abordagem utilizada por Mehtre [4], baseada em estatísticas locais, é bastante sensível ao ruído e não apresenta um manuseio simples. Além disso, tal procedimento gera um mapa de direções que contém o mesmo volume de dados do que o da imagem original, duplicando, assim, a ocupação de memória (para a segmentação da imagem é necessário também manter a informação do nível de cinza).

Uma opção vantajosa, face às características do problema em questão (obtenção do mapa de direções), é a utilização de uma rede neural dada a sua reconhecida capacidade de generalização e pouca sensibilidade ao ruído. Outro ponto a favor desta escolha é a redução da dimensão do mapa de direções aliado ao fato de não ser mais necessário manter-se a imagem original na memória, como será evidenciado mais adiante.

Para que a rede neural possa apresentar o resultado desejado, os dados de entrada na rede devem ser escolhidos de forma a criar classes com características independentes entre si. Após um exaustivo estudo das principais características presentes nas imagens (linhas, formas), verificou-se que uma análise das freqüências espaciais contidas nestes padrões fornece um conjunto de dados adequados para permitir uma separação em classes de padrões. Esta transformação dos dados de entrada da rede neural foi então realizada através da Transformada de Fourier Discreta (TFD) [5]. Além do mais, pela utilização da TFD, pode-se manipular convenientemente (minimizar) os ruídos de altas freqüências. Em especial, foi utilizado o módulo do espectro de freqüências por sua característica de invariância à translação. Outro ponto favorável à utilização da TFD é o fato de ela produzir o mesmo resultado (a menos do nível DC), tanto para imagens positivas, quanto para seus complementos (imagens negativas). Isto acontece porque os espaços entre as linhas possuem a mesma largura média das linhas. Logo, o espaço inter-linhas, conhecido como vales, também pode ser considerado um padrão de direção.

O tamanho do bloco escolhido para o cálculo da TFD foi de 8 x 8 *pixels*. Esta escolha foi baseada na largura média das linhas (4 *pixels*) e em razões práticas de implementação de um algoritmo de FFT bidimensional para realizar a TFD. Para a nossa aplicação, a utilização de um bloco maior, por exemplo, 16 x 16 *pixels*, foi descartada por causa do tamanho da rede neural resultante (levando a um maior tempo de processamento para cada impressão digital durante a operação da rede) e da maior complexidade de seu treinamento.

III. ESTRUTURA UTILIZADA

Para a realização do processo de classificação da direção principal da linha, foi escolhida uma estrutura neural *feedforward*, baseada no algoritmo *backpropagation* para seu treinamento. Para que a estrutura da rede não ficasse muito grande, foi realizada a seguinte distribuição de neurônios: 24 na camada de entrada, 8 na camada intermediária e 8 na camada de saída. A escolha dos 8 neurônios da camada intermediária ocorreu durante a fase de treinamento da rede. A estrutura foi testada com várias configurações, variando-se a camada intermediária entre 6 e 12 neurônios. O número de variáveis de entrada foi escolhido em função da TFD gerar apenas 25 valores distintos,

dispensando, assim, a utilização de todos os 64 valores obtidos na saída da TFD bidimensional. O valor correspondente ao nível DC não foi considerado na análise, pois este representa, na prática, apenas o número de *pixels* brancos (ou pretos) da imagem. Para as impressões digitais "binarizadas", cada bloco apresentará sempre um nível DC entorno de 0,5 (quantidade de *pixels* brancos aproximadamente igual a de *pixels* pretos), independente da direção da linha. Logo, este dado é irrelevante, pois não traz nenhuma informação útil para a rede neural.

As funções de transferência dos neurônios das camadas intermediária e de saída são sigmoids da forma $y = 1/(1-e^{x})$. Cada neurônio possui as sinapses ligadas às saídas de todos os neurônios da camada anterior.

A rede apresenta na saída 8 classes distintas relativas às seguintes direções: 0°, 22,5°, 45°, 67,5°, 90°, 112,5°, 135°, 157,5°. Estas classes são codificadas pelos dígitos de 0 a 7, respectivamente. O treinamento da rede foi efetuado para que tivéssemos na saída desta apenas um neurônio ativo representando cada uma dessas classes. Para a aplicação em questão, este nível de precisão foi julgado suficiente, pois, de um lado ele permite discriminar com razoável definição as direções principais das linhas que constituem as impressões digitais e do outro, aproveitar ao máximo a resolução angular (linhas, em média, com 4 a 5 *pixels* de largura) contida em cada bloco de 8 x 8 *pixels*.

IV. TREINAMENTO DA REDE

Para a realização do treinamento e avaliação (testes) da rede neural foram criados blocos de imagens binárias, de tamanho 8 x 8 *pixels*, correspondendo a cada um dos definidos padrões de direção, conforme ilustrado na Fig. 4. Estas imagens indicam a presença de linhas com 4 a 5 *pixels* de largura, representando a média da largura das linhas encontradas nas imagens "binarizadas" das impressões digitais.

Embora o módulo da TFD bidimensional seja invariante ao deslocamento, também foram utilizados, nesta etapa de treinamento, padrões com deslocamento de linhas. Esta estratégia foi empregada para possibilitar um treinamento exaustivo, utilizando-se todo o universo possível de padrões de imagens. Com isso, obteve-se um grupo de 248 padrões que resultam em 8 classes distintas quando da aplicação da TFD. Na etapa seguinte, a estas imagens foram adicionados vários níveis de ruído aleatório (inversão de pixel). Desta forma, gerou-se diversos blocos de imagens com 1, 2, até 7 *pixels* invertidos para cada padrão obtido.

Para o treinamento da rede, foram geradas várias sequências de padrões, compostas por grupos de padrões sem ruído e grupos com dois níveis de ruídos (por exemplo, 2 e 5 *pixels* de ruído por bloco). As seqüências utilizadas chegaram a ter até 1736 padrões diferentes. Assim, nesta etapa, foi selecionado, baseado no desempenho da rede, o número de neurônios que constituiria a camada intermediária. Foi também determinado o nível de ruído adequado que deixa a rede mais genérica. Para a avaliação do desempenho da rede, foi utilizada uma seqüência de blocos construída por todos os grupos de padrões com ruído, que não haviam sido utilizados durante o treinamento. Desta forma, os ajustes na etapa de treinamento foram considerados terminados quando a seqüência de teste foi totalmente classificada com 100% de acerto.

V. RESULTADOS

A validação deste processo foi realizado de maneira experimental através da implementação desta rede neural como o módulo de extração do mapa de direções de nosso sistema automático. Colocada sob teste em condições reais, a rede teve um desempenho muito bom. A Fig. 3 apresenta, de forma estilizada, o mapa de direções da impressão digital "binarizada" da Fig. 2. Deve-se aqui ressaltar, que o mapa de

direções real obtido possui um tamanho 64 vezes menor. Isto acontece pois este mapa é construído substituindo-se cada bloco de 8 x 8 *pixels* pela informação de sua direção preferencial, codificada de 0 a 7. Este mapa será reduzido ainda mais para a obtenção do esqueleto da impressão digital. Esqueleto este que terá a informação espacial disposta numa matriz de 8 x 8 elementos a ser utilizada como informação de entrada em outra rede neural, que finalmente fornecerá a classificação da impressão digital processada.

VI. CONCLUSÃO

Neste trabalho nós apresentamos uma nova forma de obtenção do mapa de direções de impressões digitais, através do emprego de uma rede neural. A decisão de se utilizar informações frequênciais como dados de entrada para a rede, mostra que várias técnicas de análise podem ser concatenadas na busca de uma solução eficiente. Colocada sob teste, em condições reais, a rede neural obtida teve um excelente desempenho na determinação do mapa de direções de impressões digitais.

Além da aplicação discutida neste trabalho, o método proposto, com pequenas alterações, pode ser utilizado para qualquer outro tipo de imagem binária que apresente características semelhantes às das impressões digitais, como por exemplo, desenhos de circuito impresso.

AGRADECIMENTO

Gostaríamos de agrader ao Prof. Luis Pereira Calôba pela inestimável ajuda prestada, principalmente, pelo incentivo à realização deste artigo.

REFERÊNCIAS

[1] L. P. Calôba, "Uma Introdução às Redes Neurais", Minicurso do 9º CBA, Vitória,ES, 1992.

[2] P. D. Wasserman, *Neural Computing: Theory and Practice*, New York: Van Nostrand Reinhold, 1989.

[3] J. A. Freeman, *Neural Networks: Algorithms, Applications, and Programming Techniques*, New York: Addison-Wesley Publishing, 1991.

[4] B. M. Mehtre and B. Chatterjee, "Segmentation of Fingerprint Images: A Composite Method", *Pattern Recognition* **22** (4), pp. 381-385 (1989)

[5] G. Zelniker, *Advanced Digital Signal Processing: Theory and Applications*, New York: Marcel Dekker, 1994.

[6] A. Oppenhien, *Signal and Systems*, New York: Practice-Hall, 1991.

Fig.1-Impressão digital adquirida (original).

Fig.2-Impressão digial "binarizada".

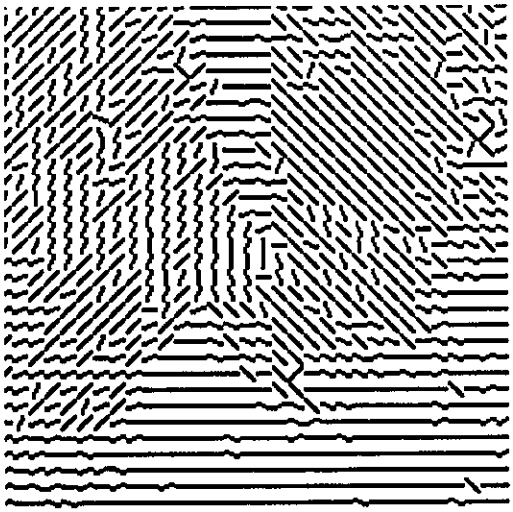


Fig.3-Mapa de direções obtido pela rede neural.

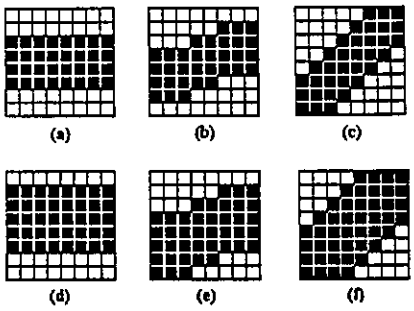
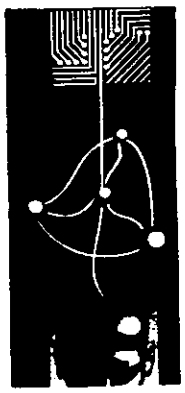


Fig.4-Exemplos de padrões de direções 0°, 22,5°, 45°. (a) (b) (c) 4 pixels e (d) (e) (f) 5 pixels.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá. 24 a 27 de outubro de 1994

RECONHECIMENTO DE PEÇAS UTILIZANDO REDES NEURAIS

Idmilson H. Sepeda Filho*
Marcelo R. Stemmer

Laboratório de Controle e Microinformática
Universidade Federal de Santa Catarina
88040-900, Florianópolis, SC, Brasil,
fax:: (0482)31-9770
e-mail: idmilson@lcmi.ufsc.br
marcelo@lcmi.ufsc.br

RESUMO

O reconhecimento de objetos por computador é uma área de grande interesse. Novos paradigmas, como redes neurais, estão sendo utilizados com o propósito de conseguir sistemas que sejam robustos e que executem esta tarefa rapidamente. Este trabalho mostra a utilização de uma rede neural *feedforward* para reconhecimento de peças mecânicas.
Palavras chave: Reconhecimento de Objetos, Redes Neurais, visão

A System for Workpiece Recognition using Neural Networks

ABSTRACT

The object recognition by computer is an area of great interest. New paradigms as neural networks are now been used to obtain sytems that are robust and solve this task faster. This paper shows the use of a feedforward neural network in mechanical workpiece recognition.
Key words: Object Recognition, Neural Networks, vision

1 INTRODUÇÃO

Visão por computador é uma área que vem recebendo muito interesse nos últimos anos. Já existem métodos para reconhecimento de padrões utilizando algoritmos convencionais de processamento de imagem.

As redes neurais surgem como um novo paradigma nesta área. Suas habilidades especiais como a capacidade de aprender por exemplos, tolerância a entradas corrompidas e reconhecimento de padrões mostram sua utilidade em aplicações de processamento de imagem.

Uma comparação entre técnicas convencionais de processamento de imagem e a técnica utilizando redes neurais é mostrada na tabela 1 [9].

As redes neurais são compostas de elementos que realizam muitas funções que são análogas as funções elementares do neurônio biológico

Além da semelhança superficial com a anatomia do cérebro, essas redes exibem algumas características deste. Por exemplo, elas aprendem por experiência. Mas apesar dessas similaridades funcionais, redes neurais artificiais estão longe de duplicar as funções do cérebro humano.

Redes neurais artificiais podem modificar seu comportamento em resposta a seu ambiente. Este fato, mais que qualquer outro, é responsável pelo interesse que vêm

* Bolsista do PICD-UFPa/CAPES

recebendo. Diz-se então que ela pode aprender (***learn***). Existe uma grande variedade de algoritmos de treinamento, todos com seus pontos fortes e fracos.

	Area	Boundary Following	Histo-grama	Redes Neurais
Tempo de processamento (segundos)	0.44	0.16	0.44	0.68 (0.28)
Precisão (%)	96	72	100	100
Orientação	-	-	Firme	Solta
Tempo de Ajuste (minutos)	10	15	10	60

Tabela 1 - Comparação de Técnicas de Processamento de imagens

Uma vez treinada, uma resposta da rede pode ser insensível à pequenas variações na sua entrada. Esta habilidade é essencial para o reconhecimento de padrões no mundo real, por causa de ruídos ou distorções de padrões (imperfeições) do mundo em que vivemos. É importante notar que os resultados das redes são obtidos a partir de sua estrutura e não pelo uso de inteligência humana embutida em alguma forma de programa de computador.

Propõe-se neste trabalho a utilização da técnica de redes neurais no reconhecimento de peças mecânicas a serem transportadas por uma esteira e identificadas por meio de uma câmera CCD (Charge Coupled Device). O sistema de reconhecimento de peças é parte de uma célula flexível de manufatura (FMC), mostrada na figura 1.

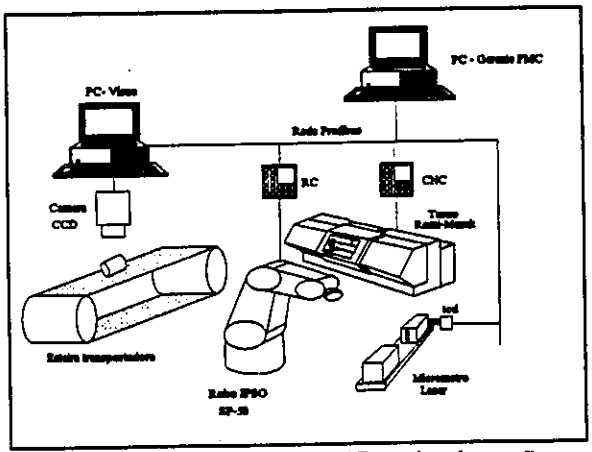


Figura 1 - Estrutura da FMC em implantação

2 ESCOLHA DE UMA ARQUITETURA DE REDE E ALGORITMO DE TREINAMENTO

Tendo em vista os objetivos propostos foi escolhida a rede counterpropagation, desenvolvida por Robert Hecht-Nielsen, para implementar o sistema de reconhecimentos de peças. Em comparação com Backpropagation, ela é de uso mais restrito, entretanto seu tempo de treinamento é bem menor. Esta característica pode ser importante, tendo em vista a tendência atual de maior diversificação e diminuição do tamanho dos lotes de peças produzidas, requerendo assim aprendizado frequente de novas imagens.

Counterpropagation é a combinação de dois algoritmos bem conhecidos: o ***self-organizing map*** de Kohonen e o ***outstar*** de Grossberg.

O processo de treinamento associa vetores de entrada com correspondentes vetores de saída. Esses vetores são números reais normalizados entre 0 e 1. Uma vez que a rede esta treinada, a aplicação de um vetor de entrada produz o desejado vetor de saída. A capacidade de generalização da rede permite que ela produza uma saída correta, mesmo quando é dado um vetor de entrada parcialmente incompleto ou parcialmente incorreto.

A figura 2 mostra uma versão simplificada da rede counterpropagation, que ilustra as características funcionais deste paradigma.

Os neurônios na camada 0 (mostrados como círculos) servem somente como ***fan-out*** e não efetuam nenhuma computação. Cada neurônio na camada 0 conecta-se com todos os neurônios na camada 1 (chamada camada Kohonen) através de um peso w_{mn}. Eles serão coletivamente referidos como a matriz de pesos **W**. Similarmente, cada neurônio na camada Kohonen (camada 1) conecta-se com todos os neurônios na camada Grossberg (camada 2) por um peso v_{np}, aqui referenciadas como a matriz de pesos **V**.

Camada Kohonen

A camada Kohonen foi treinada na sua forma mais simples, onde o "ganhador leva tudo" (*"winner-take-all"*), isto é, para um dado vetor de entrada, uma e somente uma saída dos neurônios Kohonen é logicamente 1; todas as outras são zero.

Associado com cada neurônio Kohonen esta um conjunto de pesos conectando-o à cada vetor de entrada. Como com neurônios em muitas redes, a saída NET de cada neurônio Kohonen é simplesmente a somatória das entradas ponderadas. Isto pode ser expressado como segue:

$$NET_j = w_{1j}x_1 + w_{2j}x_2 + \ldots + w_{mj}x_m$$

onde NET_j é saída NET do neurônio Kohonen j.

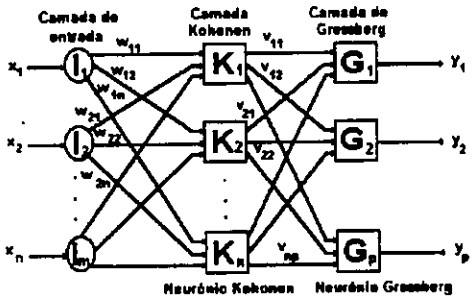


Figura 2 -Rede Counterpropagation Feedforward

O neurônio Kohonen com maior valor NET é o "ganhador". Sua saída é setada para 1; todas as outras são setadas para zero.

Camada Grossberg

A camada Grossberg funciona de maneira similar. Sua saída NET é a somatória ponderada das saídas da camada Kohonen K_1, K_2, ...,K_n, formando o vetor **K**. A saída NET de cada neurônio Grossberg é então

$$NET_j = \sum_i k_i v_{ij}$$

onde NET_j é a saída do neurônio Grossberg j.

Como a camada Kohonen é operada tal que somente uma saída NET é 1 e todas as outras são zero, somente um elemento do vetor **K** não é zero, e o cálculo é simples. De fato, a única ação de cada neurônio na camada Grossberg é colocar na saída o valor do peso que conecta-o ao único neurônio Kohonen que não é zero.

Treinamento da Camada Kohonen

A camada Kohonen classifica os vetores de entrada dentro de grupos que são similares. Isto é conseguido com o ajuste dos pesos da camada Kohonen tal que vetores de entrada similares ativam o mesmo neurônio Kohonen. Daí em diante é responsabilidade da camada Grossberg produzir a saída desejada.

O treinamento Kohonen é um algoritmo ***self-organizing*** que opera em modo não supervisionado. Por esta razão, é difícil (e não necessário) predizer qual neurônio Kohonen específico será ativado por um dado vetor de entrada. Só é necessário assegurar que o treinamento separe vetores de entrada que não sejam similares.

O neurônio com maior produto é declarado o "ganhador" e seus pesos são ajustados. Como a operação produto usada para calcular os valores de NET é uma medida de similaridade entre os vetores de entrada e peso, o processo de treinamento consiste na seleção do neurônio Kohonen cujo vetor peso é mais similar ao vetor de entrada, e torná-lo ainda mais similar, como mostra a equação abaixo.

$$w_{novo} = w_{velho} + \alpha(x - w_{velho})$$

onde:

w_{novo} = o novo valor de um peso conectando um componente de entrada x para o neurônio ganhador.

w_{velho} = o valor anterior deste peso.

A variável α é um coeficiente de taxa de treinamento que usualmente inicia próximo a 0.7 [1] e é reduzida durante o treinamento para 0.

Setando α para valores baixos reduz-se o efeito de cada passo do treinamento, fazendo o valor final uma média dos vetores de entrada para o qual ele foi treinado. Desta maneira, os pesos associados com um neurônio assumirão um valor próximo ao "centro" dos vetores de entrada para o qual aquele neurônio é o "ganhador ".

Os pesos foram setados para valores iniciais antes do início do treinamento. É prática comum em redes neurais randomizar os pesos para pequenos valores.

Treinamento da Camada Grossberg

A camada Grossberg é relativamente simples de treinar. Um vetor de entrada é aplicado, as saídas Kohonen são estabelecidas, e a saída Grossberg é calculada como na operação normal. A seguir, cada peso é ajustado somente se ele esta conectado a um neurônio Kohonen que tem uma saída diferente de zero. A quantidade de ajuste no peso é proporcional a diferença entre o peso e a saída desejada dó neurônio Grossberg para o qual ele esta conectado. Em símbolos

$$v_{ijnovo} = v_{ijvelho} + \beta(y_j - v_{ijvelho})k_i$$

onde

k_i = a saída do neurônio Kohonen i (somente os neurônios diferentes de zero).

y_j = componente j do vetor de saídas desejadas. Inicialmente β é setado em aproximada-mente 0.1 [1] e é gradualmente reduzido com o progresso do treinamento.

Com isso pode-se ver que os pesos da camada Grossberg irão convergir para valores médios das saídas desejadas, considerando que os pesos da camada Kohonen são treinados para valores médios das entradas. O treinamento Grossberg é supervisionado; o algoritmo tem a saída desejada para o qual ele é treinado. A não supervisionada operação de auto organização da camada Kohonen produz saídas em posições indeterminadas; elas são mapeadas para a saídas desejadas pela camada Grossberg.

3 PRÉ-PROCESSAMENTO DA IMAGEM

Foi utilizada uma câmera CCD para obter as imagens das peças a serem reconhecidas. Cada imagem compõe-se de 512 x 512 pixels, sendo que apenas 480 x 512 pixels são aproveitados. As imagens são em níveis de cinza que variam de 0 a 255.

Verificou-se nesta etapa que a iluminação feita sobre a peça é de grande importância. A iluminação deve ser feita de forma a não gerar sombras. A estrutura para obtenção da imagem esta mostrada na figura 3.

Desde de que seria totalmente inviável mapear diretamente cada pixel da imagem para um neurônio de entrada da rede (seriam necessários 245.760 neurônios) existe a necessidade de pré-processar a imagem.

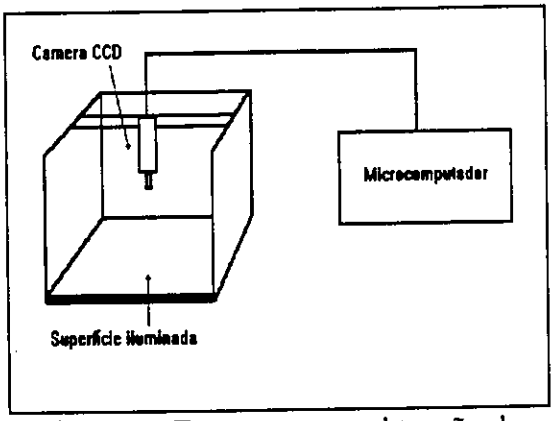


Figura 3 - Estrutura para obtenção da imagem

Para cada peça foram traçadas duas curvas de intensidade [10], uma no eixo X e outra no eixo Y, com 20 pontos cada uma. As curvas de intensidade são obtida pela soma dos níveis de cinza em cada linha e coluna da imagem., respectivamente. Desse modo, cada peça teria uma identidade própria.

4 IMPLEMENTAÇÃO E RESULTADOS

Tanto o programa para pré-processamento da imagem quanto o que implementa a rede neural foram escritos em Turbo C.

A rede neural foi implementada com 40 neurônios na camada de entrada, 235 neurônios na camada Kohonen e 13 neurônios na camada Grossberg. Foram utilizadas 13 peças, das quais foram obtidas imagens de 20° em 20°. A imagem de uma das peças utilizadas, a curva de intensidade correspondente e o comportamento do erro médio quadrático são mostrados nas figuras 4, 5 e 6, respectivamente.

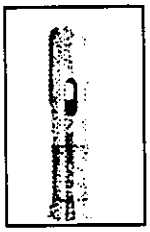


Figura 4 - Exemplo de uma peça utilizada

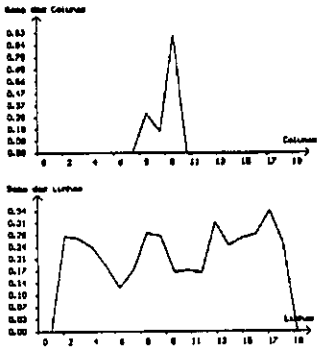


Figura 5 - Curvas de intensidade da peça exemplo

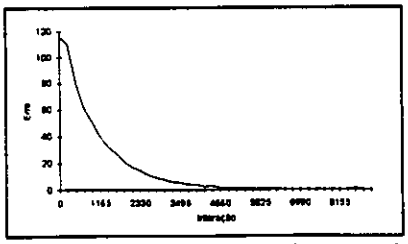


Figura 6 - Comportamento do erro durante o treinamento

Resultados experimentais obtidos com um conjunto de peças, conduziram a um tempo de treinamento relativamente baixo (aproximadamente de 40 minutos) em IBM-PC 386DX40 com, com uma taxa de acerto no reconhecimento de cerca de 86% para a rede que reconhece as peças, que após o treinamento é feito em tempo muito reduzido (da ordem de alguns milisegundos, sem considerar o tempo de leitura e digitalização da imagem).

Além do reconhecimento, o sistema informa a translação e a rotação da peça em relação ao sistema de coordenadas da câmera. Para tanto, uma nova rede foi utilizada, possuindo 40 neurônios de entrada, 156 neurônios na camada intermediária e 2 neurônios na camada de saída. Como saída desejada são fornecidos o cosseno e o seno do ângulo. No treinamento desta rede foram usadas imagens de 30° em 30° de cada peça e no modo de operação normal foi utilizada uma interpolação linear para auxiliar na precisão dos resultados. Para o treinamento foram necessárias cerca de 8.000 iterações em aproximadamente 15 minutos. A curva do erro é bastante similar àquela mostrada para a rede que reconhece as peças.

Durante o treinamento, a rede era treinada de modo a existir somente um neurônio ganhador da competição; já durante o reconhecimento os dois neurônios com maior valor de saída eram declarados ganhadores da competição. Entretanto, a somatória das saídas desses dois neurônios não pode ultrapassar 1, porque se somente um neurônio fosse declarado ganhador sua saída seria levada a 1 e os restantes seriam levados a 0. Os valores de saída de cada neurônio obtidos no final do treinamento são salvos junto com a base de conhecimento. Esses valores são aqueles obtidos com a aplicação dos padrões de treinamento, ou seja, se um neurônio foi treinado para reconhecer uma determinada peça rotacionada de 30°, por exemplo, quando da aplicação deste padrão na entrada da rede, esse neurônio terá seu maior valor de saída. Esses valores são utilizados, para dizer o quão próximo deste

valor a saída de um determinado padrão esta.

Para que se pudesse associar cada neurônio da camada Kohonen com um determinado ângulo, o treinamento desta camada deixou de ser não supervisionado e passou a ser determinístico, ou seja, padrões rotacionados de um determinado ângulo ativam um único neurônio pré-especificado.

Com esta estrutura de rede conseguiu-se uma taxa de acerto de aproximadamente 74%, e precisão de ±10°.

A translação da peça é obtida no momento da leitura da imagem, por um processo que verifica a rápida variação do nível de cinza do fundo (fundo branco) sobre o qual a peça está situada e o nível de cinza da peça.

5 CONCLUSÃO

O sistema aqui desenvolvido e implementado, possui uma taxa de acerto considerada boa, tendo em vista que outros métodos possuem taxas de acerto entre 72% e 100% [9]. Existem algumas razões para que esta taxa de acerto não seja maior. Foi utilizada uma iluminação não simétrica, ou seja, usou-se duas lâmpadas fluorescentes de cada lado da estrutura usada para obter as imagens. Como o algoritmo de pré-processamento de imagem é bastante simples e baseia-se no nível de cinza de cada pixel da imagem, esta deficiência na iluminação tende a mudar os níveis de cinza de alguns pixels que representam uma determinada área da peça sendo vista, dependendo da rotação e translação da mesma.

Outro motivo é a semelhança de algumas peças usadas no processo de treinamento, algumas delas diferem basicamente de tamanho, de maneira sútil.

Possíveis melhoramentos podem ser conseguidos, utilizando uma iluminação melhor e um algoritmo para pré-processamento que consiga extrair mais características da peça. Uma possibilidade seria a utilização de redes com inibição lateral. Esse tipo de rede pode representar padrões mais complexos. Estudos estão sendo feitos neste sentido.

BIBLIOGRAFIA

[1] - Wasserman, Philip D.: *Neural computing: theory and practice*, Van Nostrand Reinhold, 1989.

[2] - Freeman, James A. e Skapura, David M.: *Neural networks: algoritms, aplications, and programming techniques*, Addison-Wesley Publishing Company, Inc, 1991.

[3] - Doyhoff, Judith.: *Neural network architectures: an introduction*, Van Nostrand Reinhold, 1990.

[4] - Lawrence, Jeannette J.: *Untangling neural nets*, Dr. Dobb's Journal, Abril 1990.

[5] - King, Todd: *Using neural networks for pattern recognition*, Dr. Dobb's Journal, Janeiro 1989.

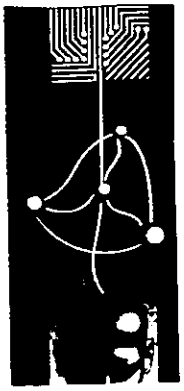
[6] - Sepeda Filho, Idmilson H. e Stemmer, Marcelo R. : *Um Sistema de Reconhecimento de Peças Baseado em Redes Neurais*, aceito para publicação nos anais do 10° Congresso Brasileiro de Automática, a realizar-se em Setembro de 1994.

[7] - Jones, William P. e Hoskins, Josiah: *Back-propagation: a generalized delta learning rule*, Byte, Outubro 1987.

[8] - Treleaven, Philip; Pacheco, Marco e Vellasco, Marley: *VLSI Architecture for Neural Networks*, IEE MICRO, Dezembro 1989.

[9] - Chien-nam, Huang; Chin-Choon, Lim e Ming, Liu C.: *Comparison of Image Processing Algorithms and Neural Network in Machine Vision Inspection*, Proceedings of the 14th Annual Conference on Computer and Industrial Engineering, Novembro 1992.

[10] - Wolf, Thomas: *Neuronen im Computer*, Revista MC, Alemanha, Abril 1990.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá. 24 a 27 de outubro de 1994

ALGORITMOS GENÉTICOS E REDES NEURAIS APLICADOS AO PROBLEMA DA ORIENTAÇÃO VISUAL EM ROBÔS MÓVEIS.

R.Glauco de Souza Rodrigues
FFCLRP - USP

CIDRA - Depto. de Física

RESUMO

Este trabalho tem por objetivo discutir a combinação de Algoritmos Genéticos com técnicas de retro propagação de erros para otimizar o problema da orientação visual em robôs móveis. A evolução genética de perceptrons multi - camadas (*perceptrons multi layers*) (GANNET)[1] foi estudada, e mostrou-se útil no projeto de perceptrons. Foi estudada também uma estrutura de retro propagação de erros, ou retroprojeção para aqueles familiarizados com os conceitos da tomografia. Chamamos essa estrutura de *Tomotron.*

INTRODUÇÃO

Os algoritmos genéticos, introduzidos por Holland na década de 70[2], foram aplicados no presente trabalho, ao problema da orientação em robôs móveis. Duas formas de aplicação foram abordadas: a evolução genética de perceptrons multi-camadas e o *Tomotron* sugerido nesse trabalho.

No caso dos perceptrons, foi considerado um sistema predador-presa confinado em um ambiente bidimensional limitado. Tanto o predador quanto a presa são modelados por perceptrons. Esse ambiente foi simulado na tela de um PC, e a aprendizagem pode ser visualizada simultânea e interativamente. O software desenvolvido em C++ para Windows permite observar na tela, o comportamento de "criaturas" que "lutam" pela sobrevivência.

Assumimos aqui, que o leitor tenha familiaridade com as idéias básicas da retro propagação (*backpropagation*) e dos algoritmos genéticos (*GA*)[3].

O uso de algoritmos genéticos para compor a estrutura de perceptrons constitui um campo de pesquisa que oferece muitos caminhos. Parâmetros como momento, razão de aprendizagem, número de sinapses, valor inicial das sinapses, alimentação para frente (*feed-forward*)[4][5] e outros podem ser codificados geneticamente.

Nesse trabalho, foram codificados: razão de aprendizagem, momento, número de camadas escondidas, número de neurônios por camada escondida, parâmetros da função de transferência e semente do gerador de números pseudo-aleatórios.

Os perceptrons foram avaliados dois a dois num esquema predador-presa. Tanto o predador como a presa foram modelados por perceptrons, sendo um treinado para caçar, e outro para escapar. Se o predador "come" a presa num dado intervalo de tempo, ele permanece e a presa é eliminada, e uma nova presa surge segundo sua probabilidade de seleção dada

pelo algoritmo genético. Porém, se a presa manter-se "viva" nesse período, o predador "morre de fome", e surge um novo predador. Deve-se observar a evolução dos perceptrons na ação de caçar e fugir, à medida que o tempo passa.

O tomotron foi inspirado na técnica de reconstrução de imagens tomográficas por retroprojeção.

Padrões visuais representados por funções de distribuição luminosa emitidas por objetos do ambiente do tomotron, são discretizados e retroprojetados sobre uma matriz W, segundo a direção dada pela reta que liga a posição do tomotron com a posição do objeto que produziu o padrão visual num dado referencial. Dessa maneira o tomotron constrói uma representação interna de sua vizinhança. Dada uma direção arbitrária, o tomotron é capaz de reproduzir, a partir de sua estrutura interna, o padrão visual associado com aquela direção. Isso será possível se houver condições para a convergência. Cada direção está associada, ao mesmo tempo, com um padrão visual e com a utilidade ou medida da performance (*fitness*) relacionada com aquele padrão no contexto da sobrevivência do tomotron naquele ambiente.

Nesse trabalho, o tomotron é designado para atingir alvos fixos dentro de seu ambiente. Dentre os vários objetos na sua vizinhança, ele caminhará na direção do maior ou do mais brilhante. Isto é implementado usando-se a área sob a curva de distribuição luminosa como medida da performance daquela direção.

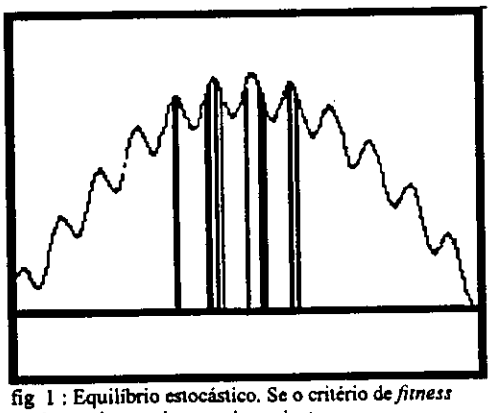


fig 1 : Equilíbrio estocástico. Se o critério de *fitness* mudar, o sistema é capaz de readaptar-se.

É possível também associar uma "rotina diária" necessária ao tomotron para suprir suas "necessidades" (por exemplo: "comer", "beber", "dormir", etc), com o equilíbrio estocástico dado pelo algoritmo genético.

Nosso interesse no tomotron deve-se ao seu potencial no que diz respeito à capacidade de armazenar um grande número de padrões. Muito deve ser feito para que essa método apresente robustez.

PADRÕES VISUAIS

Consideramos padrões visuais do tipo mostrado na figura 2.

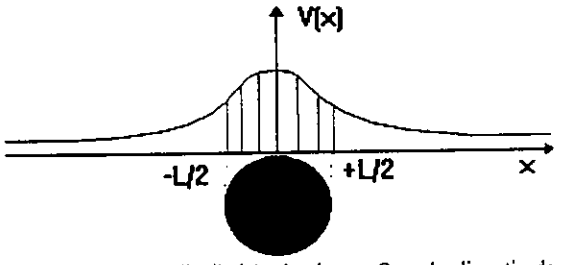


fig 2 - Função de distribuição luminosa. Quando discretizada, num intervalo fechado, transforma-se no vetor padrão visual *V*. No caso de um padrão visual bidimensional (imagem), **V** seria uma matriz.

discretizando a função de distribuição luminosa no intervalo mostrado na figura 2, obtemos o *vetor padrão visual*, assim podemos escrever:

$$V_I = [\, V_0, V_1, \ldots, V_{N-1}, V_N \,]$$

V é dado como padrão de entrada para os perceptrons e tomotrons.

Sendo L a largura do campo visual, o intervalo de amostragem é dado por:

$$I = \frac{L}{N}$$

onde *N* é o número amostras. Tanto a largura do campo visual quanto o intervalo de amostragem podem ser designados por um algoritmo genético. Assim é possível otimizar a resolução em função do tipo de

padrão a ser aprendido, economizando tempo de processamento e memória.

ALGORITMO GENÉTICO

Uma população de números inteiros é gerada pseudo-randômicamente, de maneira que cada inteiro, quando convertido em uma *string* binária, contém informações sobre a arquitetura da rede (no caso dos perceptrons). Codificar em uma *string binária* é necessário para que se possa aplicar os operadores genéticos usuais:

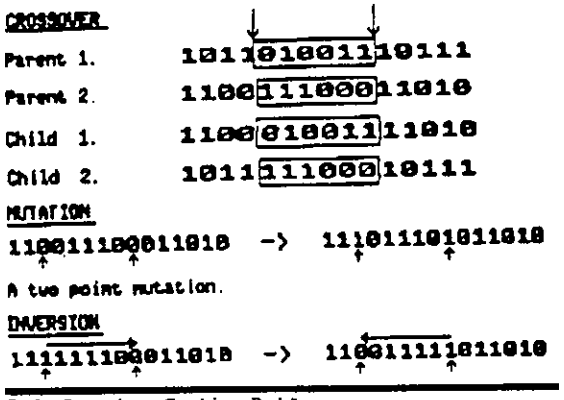


fig 3 : Operadores Genéticos Padrão.

A mutação e a inversão podem ser encaradas como uma forma de introduzir ruído para fugir da convergência prematura:

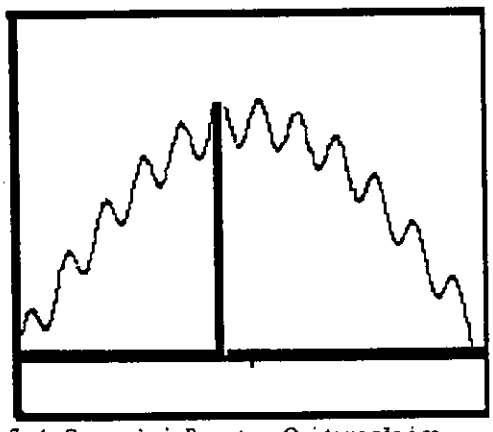


fig 4 : Convergência Prematura. O sistema não é capaz de evoluir para o máximo global.

No caso dos perceptrons, foi codificada a sua estrutura. Cada inteiro da população é convertido em uma *string* binária. Cada parâmetro do preceptron é representado por um conjunto de bits ou genes (g=1 , 0) na *string*. Esses conjuntos de genes, convertidos em inteiros separadamente vão dar os valores dos parâmetros necessários para construir a rede. Isso está ilustrado na figura 5 :

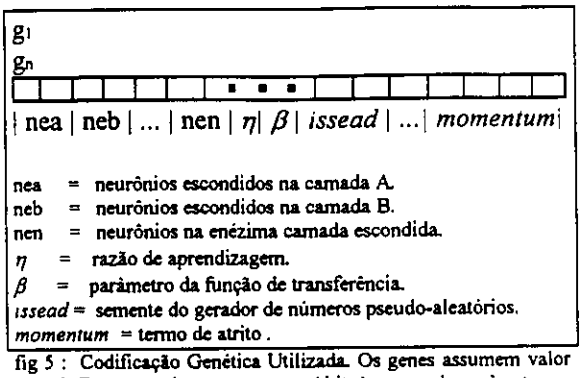


fig 5 : Codificação Genética Utilizada. Os genes assumem valor 1 ou 0. Foram usados quatro genes (bits) para cada parâmetro

Os perceptrons são avaliados dois a dois em uma caçada.

A medida da performance (v) do predador deve ser inversamente proporcional ao tempo gasto para capturar a presa, ou seja, mais apto será o predador que captura a presa no menor tempo.

A medida da performance da presa deve ser diretamente proporcional ao tempo que ela consegue fugir.

Quanto mais rápido o predador come, mais ele reproduz, quanto mais tempo vive a presa mais ela reproduz.

Se v_i é a utilidade (*fitness*) do iézimo perceptron da população, e F o número máximo de filhos por reprodução, a probabilidade de seleção e o número de filhos que ele terá serão respectivamente:

$$\mathbf{P}_i = \frac{v_i}{\sum_i v_i}$$

$$NF_i = \frac{v_i}{\sum_i v_i} \cdot F$$

Estes critérios de seleção e reprodução também é aplicado ao tomotron.

O tomotron "olha" nas direções designadas pelo ângulo θ que é

selecionado na população segundo sua probabilidade de seleção . A utilidade de cada θ é feita diretamente proporcional à área sob a curva de distribuição luminosa.

PERCEPTRONS

As redes implementadas tinham perceptrons simples como estrutura mínima (sem nenhuma camada escondida) e perceptrons de três camadas escondidas com 8 neurônios em cada uma como estrutura máxima. O algoritmo genético seleciona estruturas dentro desta faixa e computa sua utilidade possibilitando um critério para selecionar a rede que melhor resolve o problema da perseguição "visual".

As posições relativas horizontal e vertical, associadas com o padrão visual correspondente foram dadas como entrada para a rede.

Através das posições relativas, calculamos com a função arco-tangente o ângulo que especifica a direção para o predador atingir a presa, esse ângulo é usado como saída desejada (*target*).

Foi usada uma tangente hiperbólica ($\tanh(\beta x)$) como *função de transferência*. Seu parâmetro β foi modificado pelo algoritmo genético.

TOMOTRON

O tomotron é uma estrutura capaz de armazenar padrões visuais, e associar a cada um deles, uma utilidade. Quanto maior é a utilidade de um padrão, mais ele aparecerá na solução de um dado problema.

Através de um sistema de referência com origem no centro da matriz **W**, construímos um leque de retas, que gira e possibilita o cálculo das posições dos elementos da matriz, associados com cada projeção.

Para o caso bidimensional, consideramos a seguinte estrutura geométrica:

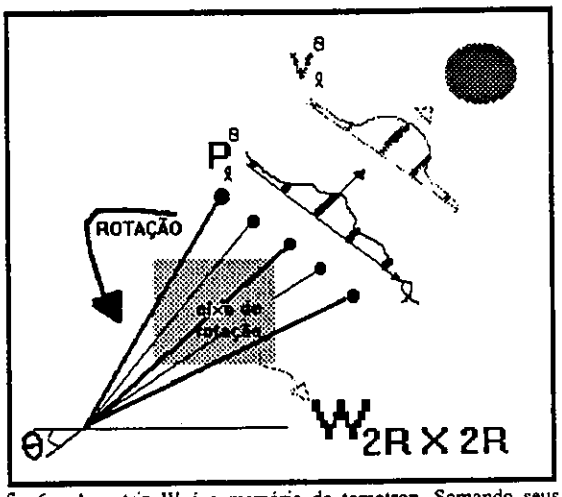


fig 6 : A matriz W é a memória do tomotron. Somando seus elementos nas direções das retas definidas pelo leque de retas da figura, obter-se o vetor de saída P_1^θ . Durante a aprendizagem, a matriz W corrigida. Ao final de um tempo t , espera-se que P_1^θ convirja para padrão visual original V_1^θ.

A equação do leque de retas é dada por:

$$y = (x - c).\mathrm{tg}\alpha$$

$$\alpha \in [\, -L\pi/360 \,,\, +L\pi/360 \,]$$

onde **c** é uma constante que deve ser maior que $2R\sqrt{2}$ (diagonal da matriz W), para que todas as projeções atravessem a matriz por completo. **L** como já foi dito, é a largura do campo visual, e deve pertencer ao intervalo :] 0 , 180 [.

Através da rotação do leque de retas sobre a matriz **W**, segundo um ângulo θ, obtém-se as posições dos elementos de matriz que devem ser atualizados . Estas posições são:

$$i = R + int(x.cos\theta + y.sen\theta)$$
$$j = R - int(-x.sen\theta + y.cos\theta)$$

$$x \in [\, -R \,,\, +R \,]$$
$$\theta \in [\, 0 \;,\; 2\pi \,]$$

De posse das projeções :

$$P_l^\theta = \sum_{ij} w_{ij},$$

o tomotron as compara com com os valores do vetor padrão visual V_l^θ relacionados com a direção designada por θ conforme a figura 6.

A regra de modificação da matriz W consiste em encontrar o desvio entre o padrão visual V_l^θ e a representação interna do tomotron, dada pela projeção P_l^θ, para o θ associado com aquele padrão.

Se considerarmos a distância euclidiana **d** entre os vetores P_l^θ e V_l^θ, para um θ em particular como uma norma para o erro, e ε o maior valor aceitável do erro, podemos considerar o padrão visual aprendido quando

$$d < \sqrt{\sum_l (P_l^\theta - V_l^\theta)^2} \quad .$$

Abaixo, construímos uma função erro conveniente a partir da distância euclidiana, e usamos o método do Gradiente Descendente para minimizar esta função :

$$E = \frac{1}{2}\left(\sum_l (P_l^\theta - V_l^\theta)^2\right.$$

A correção dos elementos da matriz **W** é

$$w_{ij} = w_{ij\,antigo} + \Delta w_{ij}$$

com

$$\Delta w_{ij} = \eta \cdot \frac{(V_l^\theta - P_l^\theta)}{r_l}$$

onde η é a razão de aprendizagem e r_l é o comprimento de caminho associado com o elemento de projeção na posição l .

O método introduz um ruído na matriz **W**. Esse ruído pode ser filtrado. Dessa forma é possível adotar um ε pequeno, aumentando a fidelidade de reprodução dos padrões visuais originais.

No instante t_0, **W** é uma matriz nula, conseqüentemente $P_l^{\theta_0}$ é um vetor nulo. Dessa forma :

$$\Delta w_{ij} = \eta \cdot \frac{V_l^\theta}{r_l}$$

é a regra de atualização de **W** no instante $t0$.

CONCLUSÃO E FUTUROS TRABALHOS

Os Algoritmos Genéticos constituem uma importante ferramenta no projeto de perceptrons.

Chegamos à conclusão de que variar apenas o número de unidades escondidas, sem modificar a estrutura de conecções, não produz modificações significativas.

Pretendemos agora concentrar esforços no estudo da evolução de perceptrons não convencionais (veja fig. 7) onde as interconecções não seguem um padrão definido.

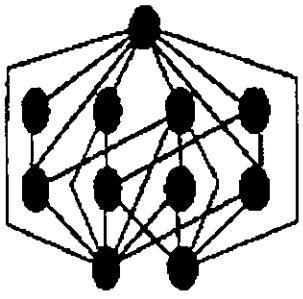


fig. 7 : Solução do problema do turno de uma espiral ( ). (razão de aprendizagem = 0.0625;momentum = 0.75)[1]

O padrão de conecção pode ser codificado geneticamente da forma mostrada na figura 8.

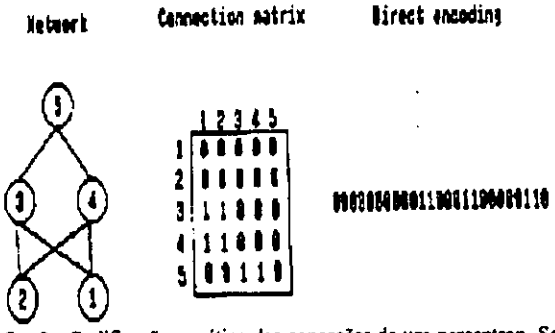


fig. 8 : Codificação genética das conecções de um perceptron. Se o valor da matriz é 1, a conecção é feita, se é zero ela não é feita.

Pretendemos implementar um tomotron tridimensional a fim de poder trabalhar com imagens.

BIBLIOGRAFIA

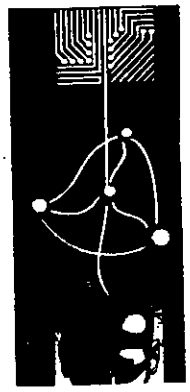
G.E.Robins, M.D. Plumbley, J.C. Hughes, F. Fallside e R. Prager ; Generation and Adaptation of Neural Networks by Evolutionary Tecniques (GANNET); Neural Comput. and Applic (1993) 1:23-31.

Holland J.H.; Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor, MI, 1975.

Goldberg D.E.; Genetic algoritms in search, optimisation and machine learning. Addison-Wesley, New York, NI, 1989.

Jones A.J.; Genetic algoritms and their applications to the design of neural networks; Neural Comput. and Applic (1993) 1:32-45.

Radcliffe J.R.; Genetic set recombination and it's application to neural network optimization.; Neural Comput. and Applic. (1993) 1:67-90



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá. 24 a 27 de outubro de 1994

Formulation of Reference Field in Neural Networks with External Input :Mechanism for Sensory Feature Integration

Kenichiro Mogi

Laboratory for Neural Networks
The Institute of Physical and Chemical Research (RIKEN)
Hirosawa 2-1, Wako-shi, Saitama, 351-01
Japan
tel +81 484-62-1111 ext. 6444
FAX +81 484-62-4697
e-mail: kenmogi@murasaki.riken.go.jp

Abstract

I apply the graphic transformation method (K. Mogi, *Phys. Rev. E* 49, 4616 (1994)) to a neural network which receive input from external neurons. I show that the state distribution of the neural network receiving input from the external neurons can be expressed as a product of the intrinsic weight and the weight of the "reference field", which is formed by the interaction between the external and "internal" neurons. The visual system can be described as a direct product of the reference field and the feature space.

I. Introduction

The problem of the integration of the visual information is one of the most important unresolved problems in neuroscience today.

The human visual system is a highly developed and complex system[1]. The light signal falling on the retina is transmitted, via the retinal ganglion cells, to the LGN (lateral geniculate nucleus) of the thalamus. The visual information is then transmitted to the primary visual cortex, where various aspects of visual information is processed in different visual areas.

In the literature, three different processes of visual information integration is currently discussed[1]. The first process involves enlarging the receptive fields of cells so that they are able to respond to, and collect information from, larger parts of the field of view. The second process occurs simultaneously with the first process and generates cells with more complex and specific properties. The third process involves the unification of signals from different sources, representing different visual attributes.

There are many theories and models aimed at explaining the mechanism of visual information integration [2]-[5].

It is one of the remarkable properties of the visual system that the third unification process occurs in such a way that the various features represented in the various "higher" visual areas are organized in the two-dimensional topographic map of the outside world. The retinotopic map of the outside world provides the frame of reference for the various visual features. In view of this aspect of visual information integration, we refer to the retinotopic map as the *reference field*. To elucidate the mechanism in which the various visual features are topographically organized

in reference to the reference field is one of the most interesting questions that can be asked about the visual system.

As the reference field for visual information integration is basically a two-dimensional retinotopic map, it is reasonable to assume that the locus where the reference field is represented in the visual area is the LGN (lateral geniculate nucleus) or the V1. Then, the question naturally arises, what is so special about the these visual areas with retinotopic representation, the LGN and V1, that they should play the role of the reference field?

A special feature of LGN and V1 is that all visual information must pass through these regions, with an exception of the superior colliculus-pulvinar pathway. Another special feature of LGN and V1 is that these areas are densely connected with the reticular nucleus of thalamus, a area generally assumed to be involved in the modulation of visual attention [6]. These special features of the LGN and V1 are expected to be closely related to the fact that the the locus of the reference field is the LGN or the V1.

Up to today, there has been no clear theoretical reasoning to backup the idea that the LGN and V1 play the role of retinotopic reference field in the visual information integration process. In this paper, we take note of the fact that the retinal ganglion cells, although a part of the visual system themselves, have a special role in the analysis of visual information. Namely, the retinal ganglion cells are special because

(1) they receive light signals from the outside world and convert them into neural firings

(2) they do not receive any back projection from the visual areas, so their activity does not depend on the internal connections and dynamics of the visual area;their activity is solely determined by the light signals falling on the retina.

It will be shown later that these properties give the retinal ganglion cells a special role in the integration of visual information.

II. Direct Product of Reference Field and Feature Space

Let us analyze what kind of mathematical structure we must have for describing the nature of visual information integration within the framework of the reference field.

Let us represent a point in the reference field as

$$\mathbf{R} = (x, y) \qquad (1)$$

Here, the variables (x, y) are any system of coordinates representing the visual field.

On the other hand, let us express the various feature categories represented in the visual cortex as

$$\mathbf{F} = (f_1, f_2, \ldots, f_p) \qquad (2)$$

Here, it is implied that the suffices represent the categories of features represented in the visual system, such as color, shape, border, texture, etc. A particular value of the variables f_1, f_2, ..., f_p would then describe a particular feature of these categories.

Then it is clear that representing the various features **F** within the frame work of reference field **R** means that we have a direct product of these fields **R x F**, namely

$$\mathbf{R} \times \mathbf{F} = (x, y) \times (f_1, f_2, \ldots, f_p) = (x, y, f_1, f_2, \ldots, f_p)$$

It is to be noted that the weight distribution of the product space **R x F** can be formally written as the product of the weights for the space **R** and **F** as

$$W(\mathbf{R} \times \mathbf{F}) = W(\mathbf{R})W(\mathbf{F}) \qquad (3)$$

Here, it should be noted that as the features in the feature space **F** are generated as

a function of the reference field **R**, the weight expression for these areas are not completely independent. This fact will be expressed mathematically that there are some common elements in the expressions of **F** and **R**.

The question arises how we can establish a direct product of the kind depicted above in a neural network receiving and analyzing information from the outside world, as in the case of the visual cortex.

In the next section, we show that the existence of "external neurons" (which only project to the visual neural network, and do not receive a back projection) leads to a representation of the visual information in terms of the direct product of the reference field **R** and feature space **F**.

III. Neural Networks with External Input

In this section, we study the properties of neural networks with external neurons. We assume that the network has stochastic dynamics (the Boltzmann machines[7]).

The network consists of N "internal" neurons, which are mutually connected *via* synapses. At a given instant, each neuron takes a value of either *0* (non-firing state) or *1* (firing state).

We represent by w_{ij} the strength of input from the j th neuron to the i th neuron.

We assume that

$$w_{ii} = 0$$

In addition to the above network of the "internal" neurons, we assume that there are T "external" neurons $r_1, r_2, ..., r_T$ which represent sensory signal coming from the outside world.

In the case of visual perception, the "internal" neurons can be considered to be the visual cortical cells (including cells in the LGN of the thalamus) and "external" neurons can be considered to represent the retinal ganglion cells.

The state of the total ("internal" + "external") population of neurons would be represented by a vertex in the $N+T$-dimensional hypercube.

$$(S,R) = (s_1,.....,s_N,r_1,.....,r_T) \in \{0,1\}^{N+T}$$

It is assumed that the connections between the "external" neurons and "internal" neurons are uni-directional. Specifically, although "external" neurons project to the "internal" neurons with non-zero weight, there are no back projection from the "internal" neurons to the "external" neurons. At the same time, we assume for simplicity that there are no mutual connections between the "external" neurons. We express the connection from the jth "external" neuron to the ith "internal" neuron as α_{ij} $(i=1,2,...,N;j=1,2,...,T)$.

It is to be noted that because of the above conditions, the states of the "external" neurons are determined solely by the nature of the signal from outside (*i.e.*, light signal falling on the retina), and not by the properties of the "internal" network of neurons.

In following discussions, we regard the states of the "external neurons" as given, and study the properties of the network of "internal" neurons under that constraint. We indicate a particular state of the "internal" neurons in $\{0,1\}^N$ as S^k, where $k=1, 2, ..., 2^N$.

Let us assume that the neurons are randomly chosen and called to update [7] with rates of activation $F(i)$, where i denotes the index of the neuron that is called to update. The rate of transition from the state S^q to the state S^p can be written as

$$K(S^p,S^q) = F(i_c)\frac{1}{1+e^{-\beta(1-2s_{i_c}^q)U_{i_c}(S^q)}} \quad (4),$$

where i_c denotes the index of the neuron that is called to update. $U_i(S)$ is the input signal on the i th neuron given as

$$U_i(S)=\sum_{j=1}^{N} w_{ij}s_j+\sum_{j=1}^{T}\alpha_{ij}r_j-\theta_i \tag{5}$$

where θ_i represents the threshold.

We assume that only one neuron changes its state at a particular time of transition (the serial update method). Namely, we have the relations

$$\begin{aligned} s_{i_c}^q &= 1-s_{i_c}^p \\ s_i^p &= s_i^q \quad (i\neq i_c) \end{aligned} \tag{6}$$

The evolution of the system is then described by the Master equation

$$\frac{\partial\rho(S^p,t)}{\partial t}=\sum_{S^q}\left(K(S^p,S^q)\rho(S^q)-K(S^q,S^p)\rho(S^p)\right) \tag{7}$$

where $r(S)$ is the probability distribution for the state S.

Using the relations

$$\sum_{j=1}^{T}\alpha_{i_c j}\left(s_{i_c}^q-s_{i_c}^p\right)r_j=\sum_{i=1}^{N}\sum_{j=1}^{T}\alpha_{ij}\left(s_i^q-s_i^p\right)r_j$$

We can then show that the rate constants satisfy the relations[9]

$$\frac{K(S^p,S^q)}{K(S^q,S^p)}=e^{-\beta(E'(S^p)-E'(S^q)+d(S^p,S^q))} \tag{8}$$

where $E'(S^p)$, $E'(S^q)$ are the effective equilibrium energy values defined as the sum of the equilibrium energy value and a term representing the contribution of the external neurons

$$E'(S)=E(S)-\sum_{i=1}^{N}\sum_{j=1}^{T}\alpha_{ij}s_i r_j \tag{9-A}$$

The equilibrium energy $E(S)$ for the internal neurons is given as

$$E(S)=-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}w_{ij}s_i s_j+\sum_{i=1}^{N}s_i\theta_i \tag{9-B}$$

The "asymmetric energy term"

$$d(S^p,S^q)=-\frac{1}{4}\sum_i\sum_j\left(w_{ij}-w_{ji}\right)\left(s_i^p-s_i^q\right)\left(s_j^p+s_j^q\right) \tag{10}$$

is a measure of the deviation from the equilibrium.

In a steady state, the state distribution $r(S)$ is given by the balance equation

$$\sum_{S^q}\left(K(S^p,S^q)\rho(S^q)-K(S^q,S^p)\rho(S^p)\right)=0 \tag{11}$$

The normalization condition is

$$\sum_S\rho(S)=1 \tag{12}$$

The steady state distribution of the Boltzmann machine can be obtained as the solution for equations (9), (11), and (12).

In order to see the effect of the input of external neurons on the state distribution of the neural network, we apply the graphic transformation method[8],[9].

We express the spanning in-trees which have a particular state S as the sink as $G_S(m)$ $(m=1,2,3...n_g)$, where n_g is the number of spanning in-trees of the hypercube $\{0,1\}^N$

The solution for the balance equation can be given by the graphic method[10] as

$$W(S)=\sum_{m}\prod_{(S^p,S^q)\in G_S(m)}K(S^p,S^q) \qquad (13)$$

where

$$\prod_{(S^p,S^q)\in G_S(m)}K(S^p,S^q) \qquad (14)$$

represents the product of the rate constants corresponding to the directed edges of the spanning in-tree $G_S(m)$. The term (S^p,S^q) represents the ordered pair of states corresponding to the directed edges of the spanning in-tree $G_S(m)$.

By applying the graphic transformation method, we can transform the weight $W(S)$ as

$$W(S)=e^{-\beta E'(S)}D(S) \qquad (15)$$

where S^O is the standard state for normalization taken arbitrarily, and $P_{SS^O(m)}$ is the path in $G_S(m)$ connecting the states S^O and S. D(S) is a measure of deviation from the equilibrium given as

$$D(S)=\frac{\sum_{m} e^{-\beta\left(\sum_{(S^p,S^q)\in P_{SS^O}(m)} d(S^p,S^q)\right)}\prod_{(S^p,S^q)\in G_{S^O}(m)}K(S^p,S^q)}{\sum_{m}\prod_{(S^p,S^q)\in G_{S^O}(m)}K(S^p,S^q)} \qquad (16)$$

For a symmetric $(w_{ij}=w_{ji})$ network, we have

$$D(S)=1$$

We now note an important property of the weight representation (15). Namely, we can write (15) alternatively as

$$W(S)=e^{-\beta\left(-\sum_{i=1}^{N}\sum_{j=1}^{T}\alpha_{ij}s_i r_j\right)}w(S) \qquad (17)$$

where

$$w(S)=e^{-\beta E(S)}D(S) \qquad (18)$$

is the weight representing the internal neurons.

We have seen in section II that in order to integrate the various features represented in the cortex within the framework of the retinotopic reference field, we would need a direct product of the reference field **R** and feature space **F**. We have now demonstrated that the state distribution of the visual system can be described by the product of two elements, one representing the reference field **R**

$$W(\mathbf{R})=e^{-\beta\left(-\sum_{i=1}^{N}\sum_{j=1}^{T}\alpha_{ij}s_i r_j\right)}$$

and one representing the feature space **F**

$$W(\mathbf{F})=w(S)$$

Therefore, we conclude that the weight distribution of the visual system can indeed be written as a product of weights for the reference field **R** and the feature space **F**.

IV Conclusions

In our theory, the external neurons play a significant role in formulating the reference field. In biological visual systems, the retinal ganglion cells correspond to the "external" neurons. The role played by these "external" neurons is interesting. Although they provide the input to the visual system, they do not receive corresponding projection back from the visual system. Therefore, the state of the external neurons are determined solely by the visual environment, and are not influenced by the internal conditions of the visual network. The state of the external neurons represent the "constraint" imposed upon the visual system by the outside world.

It is interesting to note that the "reference field", the frame of reference for the various features represented in the visual system, is formed through the interaction between the "external" neurons and the "internal" neurons. In this way, the space structure of the outside world is "copied" into the visual cortex, providing us with an internal "model" of the outside world.

From our model, we can conclude that the geometrical properties of the "reference field" is determined by the geometrical structure of the "external" neurons (the retinal ganglion cells) and those neurons receiving input from them (the LGN cells). It is known that in the LGN and V1, the retinotopic map of the retina is represented faithfully, with the modification that the central area of vision around fovea are over-represented. In our conscious perception, we do see the world as a collection of various visual features organized in a retinotopic framework. Therefore, the prediction of our model that the reference field is defined by the interaction between the retinal ganglion cells and the LGN cells is consistent with our experience.

It is to be noted that the disembodiment of the visual system into two parts **R** and **F** is unique. Due to the existence of the asymmetric term $D(S)$ in (18), it is not possible to express the weight for the internal visual system as a product of two subsystems.

We did not discuss the role of attention. In the literature, attention is usually held responsible for the integration of visual features. We proposed an alternative mechanism, in which the visual features are integrated by assignment of the various features to a particular locus of the retinotopic field. Compared to the attentional mechanism of integration, such a integration mechanism is more stable and can be conducted in a parallel manner. We consider such a pre-attentive mechanism of visual information integration as an essential feature of visual cognition.

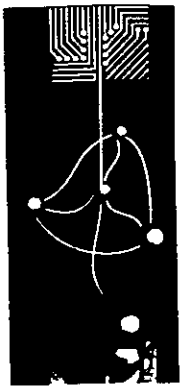
Acknowledgements

We would like to thank Horace Barlow, Tomoki Fukai, Yoshihide Tamori, Shigeru Tanaka for helpful discussions. We are grateful to Masao Ito for valuable advises and encouragement.

References

[1] Zeki, S. in *A Vision of the Brain* (Blackwell Scientific Publications) (1993).
[2] H.B. Barlow & J.D. Mollon in *The Senses* (Cambridge University Press) (1982)
[3] Damasio, A. R. *Seminars in The NeuroSciences* **2**, 287-296 (1990)
[4] Gray, C. M., König, P., Engel, A. K. & Singer, W. *Nature* **338**, 334-337. (1989)
[5] Malsburg, C. von der & Buhmann, J. *Biol. Cybern.* **67**, 233-242. (1992)
[6] Crick, F. *Proc. Natl. Acad. Sci. USA* **81**, 4586-4590. (1984)
[7] D. H. Ackley, G. E. Hinton and T. J. Sejnowski, *Cognitive Sci.* **9**, 147 (1985).
[8] K. Mogi *J. theor. Biol.* **162**, 337-352 (1993).
[9] K. Mogi *Physical Review E* **49**, 4616-4626 (1994)
[10] E.L. King, and C. Altman, *J. Phys. Chem.* **60**, 1375 (1956).

Anotações



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

CONTROLE I

Anotações

1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

REDES NEURAIS APLICADAS AO ACIONAMENTO DE MOTOR DE INDUÇÃO

Walmir Matos Caminhas, DEE/UFMG Márcio Luiz Andrade Netto, FEE/UNICAMP
Pyramo Pires da Costa Jr., DEE/UFMG Hermano M.F. Tavares , FEE/UNICAMP
FEE/UNICAMP
E-MAIL: DEPTO@DENSIS. FEE. UNICAMP. BR
DENSIS - FEE/UNICAMP - Caixa Postal 6101
13.081-970 - Campinas - SP

RESUMO

Neste trabalho é estudada a aplicação de **redes neurais** no acionamento do motor de indução trifásico de rotor em gaiola. A rede neural é utilizada para identificar e compensar as variações da resistência e indutância do rotor da máquina. Tal compensação se faz necessária quando se utiliza a técnica de orientação indireta de campo para o controle de velocidade, que é o caso deste trabalho. Com relação ao treinamento da rede é usado o método de programação não linear de Broyden-Fletcher-Goldfarb-Shanno (BFGS), conjuntamente com o algoritmo de Back-Propagation. Também é proposta uma função, para ativação dos neurônios, quociente de polinomios ordem, que aproxima-se da tangente hiperbólica (com a vantagem de ser calculada em um tempo de processamento cerca de 5 vezes menor). A partir de simulações digitais é feita uma análise do desempenho do acionamento frente às variações dos parâmetros do rotor.

1 - INTRODUÇÃO

Quando se aplicam as técnicas de controle vetorial o motor de indução, a velocidade variável, apresenta melhor desempenho que um motor corrente contínua de mesma potência (Bose, 1986). Um dos métodos de controle vetorial mais popular é o de orientação indireta de campo. Em tal método a determinação da posição do vetor fluxo rotórico é feita pela integração da soma de duas grandezas: a velocidade elétrica do rotor e a velocidade de escorregamento. Esta última requer em seu cálculo o valor exato da constante de tempo elétrica do circuito de rotor. Portanto, este método é sensível às variações dos parâmetros que compõem esta constante de tempo, e tais variações devem ser compensadas de modo a se preservar a qualidade do desempenho do acionamento. Este trabalho apresenta uma técnica de identificação e compensação destas variações, baseada em **redes neurais** (Zuben, 1993). Através de simulações digitais é feita uma análise do desempenho do sistema de acionamento (composto pelo inversor PWM de corrente, motor de indução e carga mecânica) frente às variações da constante de tempo elétrica do rotor.

2 - DESCRIÇÃO DO SISTEMA

O diagrama de blocos da figura 1 ilustra o método indireto de orientação de campo, aplicado ao acionamento de um motor de indução alimentado por um inversor de corrente (Bose, 1986).

O cálculo do valor da velocidade de escorregamento de referência, ω_s^*, é função das referências da constante de tempo elétrica do rotor, Tr, e das correntes $I_{ds}^{*(e)}$ e $I_{qs}^{*(e)}$. A operação é feita pelo bloco denominado "calcula ω_s^*" e é

baseada no modelo do motor de indução com campo orientado (Bose, 1986). É a velocidade de escorregamento de referência que irá promover a orientação de campo na máquina. A posição angular do vetor fluxo rotórico, ρ, é usada para obter as correntes de referência do estator. Estas correntes de referência uma vez comparadas com as correntes reais (medidas) fornecem os sinais de comando do inversor de corrente transistorizado (controle por valores extremos ou histerese de corrente).

Como visto no diagrama da figura, no método de controle vetorial por orientação indireta de campo, a posição do vetor fluxo rotórico é determinada a partir do valor da velocidade de escorregamento. Esta velocidade requer no seu cálculo o valor da constante de tempo de circuito de rotor, Tr. Como os parâmetros que compõem esta constante de tempo variam, o método indireto de orientação de campo é sensível à estas variações. A resistência de rotor varia com a temperatura e frequência e a indutância varia com a saturação magnética. Com um valor incorreto da velocidade de escorregamento, resultado de um valor de Tr incorreto no calculador de escorregamento, o desacoplamento entre conjugado e fluxo não é conseguido (Bose, 1986).

Para garantir o desacoplamento entre o fluxo e o conjugado eletromagnético, torna-se necessária a implementação de um método de compensar as variações da constante de tempo rotórica. Vários métodos que se denominam "métodos de identificação da constante de tempo de rotor" têm sido propostos ultimamente (Caminhas e outros, 1990). A maioria deles é baseada na comparação entre grandezas que contém informação da amplitude do fluxo rotórico, tais como potência reativa, força eletromotriz e outras, estimadas através das variáveis de controle e as estimadas através das grandezas reais medidas no motor (correntes e tensões). Desta comparação resulta um sinal de erro que promove a devida correção na constante de tempo de referência do circuito de rotor. Zuben (1993) propôs um método alternativo que é a utilização de redes neurais para este fim. Esta idéia é a explorada neste trabalho.

A partir das correntes $i_{qs}^{(e)}$, $i_{ds}^{(e)}$ e ω_r, que são as entradas, a rede neural fornece na saída o valor da constante de tempo rotórica, Tr, que será utilizado pelo bloco "calcula ω_s^*", da figura 1. As correntes $i_{qs}^{(e)}$ e $i_{ds}^{(e)}$ são obtidas a partir de Ias, Ibs e Ics após a transformação de ABC para dq0[e].

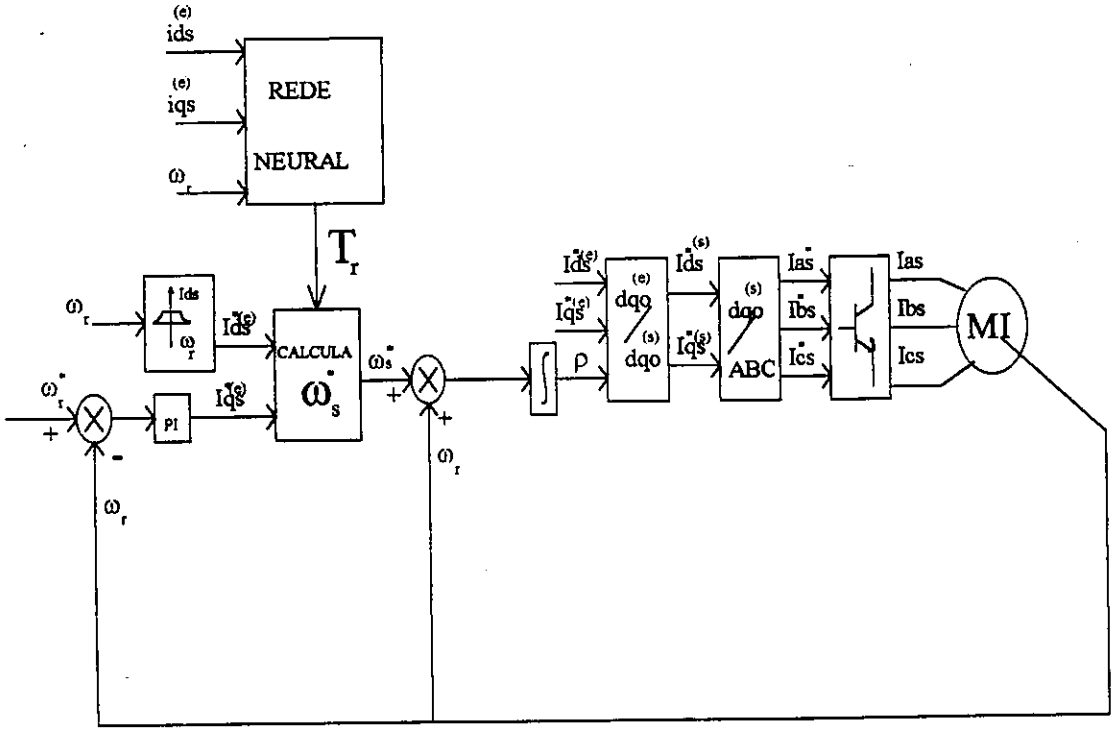


Fig. 1 - Diagrama de blocos do método indireto orientação de campo

3 - O TREINAMENTO DA REDE

A topologia da rede usada, em camadas, é mostrada na figura 2. Para simplificar o desenho, não foram mostradas todas as conexões entre os neurônios de saída e os de entrada. O treinamento consiste em ajustar os pesos sinápticos de modo a obter o menor erro quadrático. O erro é usado para atualizar estes pesos.

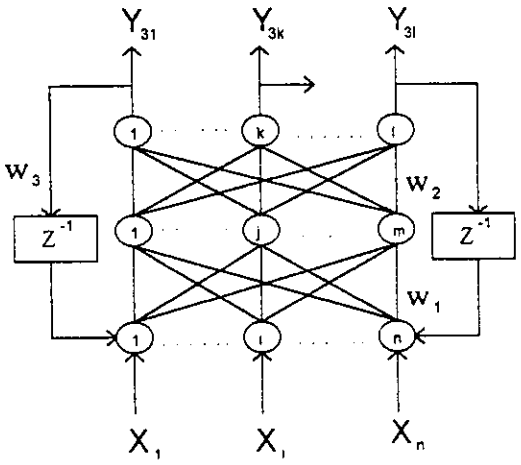


Fig. 2 - Topologia da rede neural utilizada

Sendo d_k o valor desejado, o erro quadrático é dado por:

$$E = \frac{1}{2}\left(\sum_{k=1}^{l} \left(d_k - y_{3_k}\right)^2 \right) \qquad (1)$$

onde

$$y_{3k} = f\left(w_{2jk}, w_{1ij}, w_{3ki}\right)$$

A rede utilizada neste trabalho possui 3 entradas (n=3), 15 neurônios na camada escondida (m=15) e 1 saída (l=1).

O treinamento consiste em:

$$\text{(P)} \quad \begin{aligned} & Min.E = E(\tilde{x}) \\ & s.a. \\ & \tilde{x} \in \Re^p \\ & p = n.m + m.l + l.n \end{aligned} \qquad (2)$$

onde

$$\tilde{x} = \begin{pmatrix} \tilde{x}_{W_2} \\ \cdots \\ \tilde{x}_{W_1} \\ \cdots \\ \tilde{x}_{W_3} \end{pmatrix}$$

e

$$\tilde{x}_{W_2} = \begin{pmatrix} w_{2_{11}} \\ \cdot \\ w_{2_{jk}} \\ \cdot \\ w_{2_{ml}} \end{pmatrix} ; \quad \tilde{x}_{W_1} = \begin{pmatrix} w_{1_{11}} \\ \cdot \\ w_{1_{ij}} \\ \cdot \\ w_{1_{nm}} \end{pmatrix} ; \quad \tilde{x}_{W_3} = \begin{pmatrix} w_{3_{11}} \\ \cdot \\ w_{3_{ki}} \\ \cdot \\ w_{3_{ln}} \end{pmatrix}$$

Como a função erro $E(\tilde{x})$ é não linear, (P) é um Problema de Programação Não Linear Irrestrito.

3.1 - A Programação Não Linear

A programação não linear caracteriza-se por não possuir um método geral de resolução dos seus problemas. São muitos algoritmos e quase sempre voltados para problemas específicos, presos à características diversas tais como continuidade, diferenciabilidade de primeira e/ou de segunda ordem, e outras. Uma outra particularidade é a inexistência de critérios absolutos para comparação entre os algoritmos existentes (Mateus, 1986).

Neste trabalho é utilizado o método de **Broyden-Fletcher-Goldfarb-Shanno (BFGS)** para resolver o problema (P), (Mahey, 1986). Este método é intermediário entre a simplicidade do método de Gradiente e a rapidez do método de Newton. Substitui o cálculo da hessiana no método de Newton, por um processo iterativo finito, utilizando apenas derivadas de primeira ordem.

No método BFGS a direção de busca $\tilde{h}^k$ é calculada por:

$$\tilde{h}^k = -(\tilde{D}^k)^{-1}.\tilde{\nabla}E(\tilde{x}^k) \qquad (3)$$

onde

$$\tilde{D}^{k+1} = \tilde{D}^k + \frac{\Delta\tilde{g}.\Delta\tilde{g}^T}{\Delta\tilde{x}^T.\Delta\tilde{x}} - \frac{\tilde{D}^k.\Delta\tilde{x}.\Delta\tilde{x}^T.\tilde{D}^k}{\Delta\tilde{x}^T.\tilde{D}^k.\Delta\tilde{x}}$$

sendo

$$\Delta\tilde{x} = \tilde{x}^{k-1} - \tilde{x}^{k}$$

$$\Delta\tilde{g} = \tilde{\nabla}E\left(\tilde{x}^{k+1}\right) - \tilde{\nabla}E\left(\tilde{x}^{k}\right)$$

A matriz $\tilde{D}^0$ (condição inicial) pode ser qualquer matriz definida positiva, normalmente toma-se como igual a matriz identidade.

A convergência exige $\tilde{D}^k$ definida positiva, para assegurar o decrescimento da função erro, ou seja, que $E(\tilde{x}^{k+1}) < E(\tilde{x}^{k})$. Para garantir isto, entre várias alternativas, escolhemos uma, proposta por Powell, que consiste em substituir $\Delta\tilde{g}$ por $\Delta\tilde{g}'$ (Saldanha, 1992).

$$\Delta\tilde{g}' = \theta.\Delta\tilde{g} + (1-\theta).\tilde{D}^k \qquad (4)$$

onde

$$\theta = \begin{cases} 1 & se \quad \Delta\tilde{x}^T.\Delta\tilde{g} \geq 0{,}2.\Delta\tilde{x}^T.\tilde{D}^k.\Delta\tilde{x} \\ \dfrac{0{,}8.\Delta\tilde{x}^T.\tilde{D}^k.\Delta\tilde{x}}{\Delta\tilde{x}^T.\tilde{D}^k.\Delta\tilde{x} - \Delta\tilde{x}^T.\Delta\tilde{g}} & c.c. \end{cases}$$

Deve-se notar que, para $\theta \neq 1$, temos $\Delta\tilde{g} \neq \Delta\tilde{g}'$.

Calculada a matriz $\tilde{D}^k$, seguramente definida positiva e simétrica, pode-se obter a direção $\tilde{h}^k$ por (5). Para tanto pode-se usar método de fatoração de Cholesk, com número de operações (da ordem $O(n^2)$) que é mais eficiente que a inversão $\tilde{D}^k$ (Mahey, 1987).

$$\tilde{D}^k.\tilde{h}^k = -\tilde{\nabla}E\left(\tilde{x}^k\right) \qquad (5)$$

3.2 - O Algoritmo de treinamento

Para atualizar os pesos sinápticos da rede, foi utilizado o algoritmo de "Back-Propagation" conjuntamente com o método BFGS. O gradiente do erro quadrático com relação aos pesos é dado por:

$$\tilde{\nabla}E = \begin{pmatrix} \tilde{\nabla}E_{\tilde{W}_2} \\ \tilde{\nabla}E_{\tilde{W}_1} \\ \tilde{\nabla}E_{\tilde{W}_3} \end{pmatrix} \qquad (6)$$

sendo que o cálculo das componentes dos vetores é feito de acordo com algoritmo de "Back-Propagation" (Widrow & Lehr, 1990; Kosko, 1991)

3.3 - A função de Ativação

Para a implementação prática da técnica aqui estudada, um fator bastante importante é o tempo de processamento da rede neural. Grande parte deste tempo é gasto no cálculo da função de ativação dos neurônios, que normalmente é utilizada a tangente hiperbólica (tgh(γx)). Com objetivo de tentar reduzir este tempo, a tangente hiperbólica foi aproximada pela função:

$$f(x) = \begin{cases} \dfrac{a-b}{a+b} & se \quad |x| \geq \beta \\ 1 & se \quad x > \beta \\ -1 & se \quad x < -\beta \end{cases} \qquad (7)$$

onde

$$\begin{cases} a = (x+\beta)^2 \\ b = (x-\beta)^2 \end{cases}$$

Mostra-se sem dificuldade que esta função é de classe C^1.

A equação (7) mostra que para calcular o valor de *f(x)* são necessárias três operações de multiplicação/divisão e quatro de soma/subtração, quando o valor absoluto do argumento é menor que β e somente uma multiplicação se for maior. Neste termos tem-se uma função cujo tempo de processamento é da ordem de cinco vezes menor que o tempo de cálculo da tangente hiperbólica, função bastante utilizada em redes neurais, e com resultados praticamente iguais, para os casos analisados. Um estudo comparativo entre as duas funções é mostrado nas figuras 3 e 4. Na figura 3 são mostradas as curvas da tangente hiperbólica (com γ=2.2) e de f(x) (com β=2). Já na figura 4 é mostrada a comparação entre os tempos de processamento das duas funções, no cálculo de N pontos funcionais.

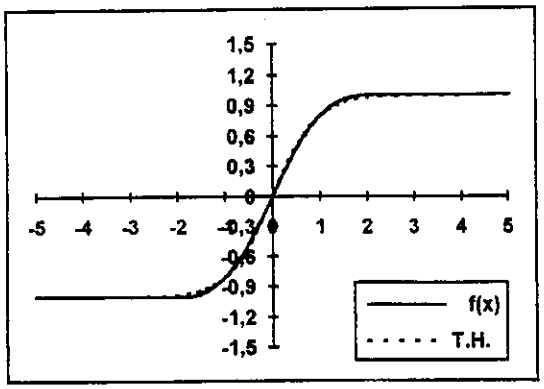


Fig. 3 - Tangente Hiperbólica e f(x).

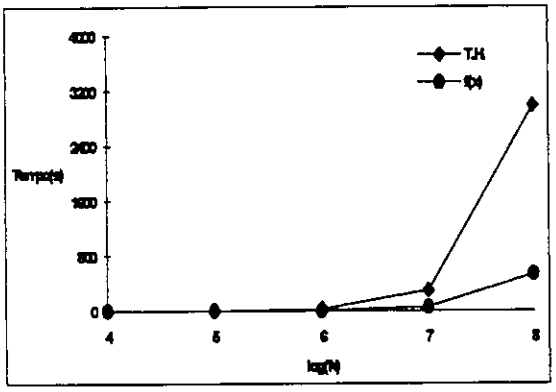


Fig. 4 - Gráfico do tempo de processamento da Tangente hiperbólica e f(x)

4 - RESULTADOS DE SIMULAÇÕES

Para simulação foram usados dados de um motor de 2 C.V., cujos parâmetros são mostrado a seguir. O programa de simulação resolve as equações diferenciais pelo método de Runge Kutta de quarta ordem. O período de integração usado foi de 20 μs e o de amostragem, para malha de controle de velocidade foi de 2 ms. Já o período de amostragem para o cálculo de Tr foi de 40 ms.

Dados da Máquina de Indução utilizada nas simulações:

- Potência = 2 HP
- Resitência do estator = 1.5 Ω
- Resitência do rotor = 1.6 Ω
- Indutância própria do estator = 0.108 H
- Indutância própria do rotor = 0.115 H
- Indutância mútua estator/rotor = 0.098 H
- Momento de inércia = 0.016 kg.m^2
- Número de pólos = 4

Uma vez treinada, a rede foi usada para identificação da constante de tempo elétrica do rotor. A figura 5 mostra as curvas das constante de tempo real e a estimada pela rede em função do tempo. Como mostrado a rede consegue identificar Tr com um erro muito pequeno, isto faz com que o sistema não perca o desacoplamento entre o fluxo rotórico e o conjugado eletromagnético, fato que fica evidenciado (figura 6) pela curva do fluxo rotórico de eixo "q", para o sistema com compensação, (FqrCc), permanecendo praticamente em zero. Nesta figura apresentamos também este mesmo fluxo, sem compensação, FqrSC, deixando claro o acoplamento entre fluxo e conjugado. A amplitude do fluxo rotórico assume um valor que é praticamente o nominal quando é feita a compensação, figura 7. Isto mostra o bom desempenho da estratégia de identificação adotada.

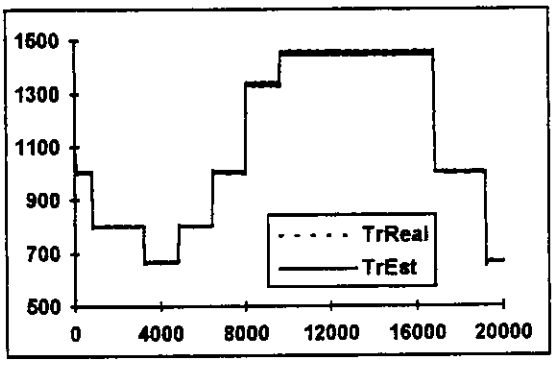


Fig. 5 - Gráfico da constante de tempo elétrica do rotor identificada (em p.u.) em função do Tempo (ms).

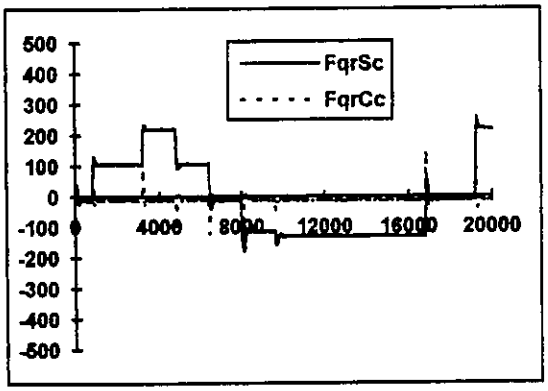


Fig. 6 - Fluxo rotórico de eixo "q" (em mWb) em função do Tempo (ms)

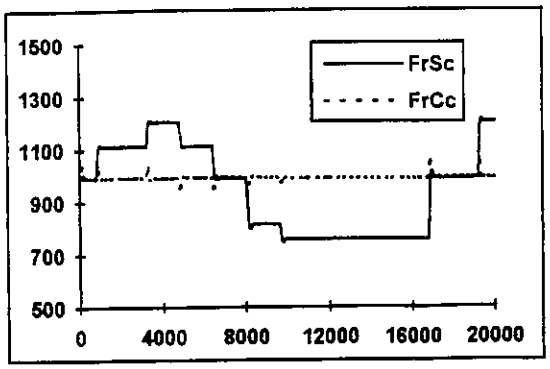


Fig. 7 - Amplitude do fluxo rotórico (em mWb) em função do Tempo (ms)

5 - CONCLUSÃO

Neste trabalho foi visto que um erro no valor adotado para a constante de tempo elétrica do rotor de referência influencia tanto no comportamento dinâmico quanto no regime permanente da máquina. Com o objetivo de solucionar o problema foi analisado um método de identificação e compensação de Tr, baseado em **Redes Neurais**. Foram utilizados o algoritmo de "Back-Propagation" conjuntamente com o método BFGS para o treinamento da rede. Foi proposta uma função tipo polinomial para ativação dos neurônios da rede. Os resultados obtidos mostram o bom desempenho da estratégia aqui apresentada na identificação e compensação de Tr.

8 - REFERÊNCIAS BIBLIOGRÁFICAS

Bose, B.K. (1986)."Power Eletronics A.C. Drives", Prentice Hall.

Caminhas, W.M.; Menezes, B.R.; Tribuzi, A.R.; Silva, S.R. (1990). "Influência, Identificação e Compensação das Variações dos Parâmetros do Motor de Indução no Controle Vetorial Indireto". In: CONGRESSO BRASILEIRO DE AUTOMÁTICA, 8, 1990, Belém-Pará. Anais Belém: Editora Sociedade Brasileira de Automática, p. 936-941.

Kosko, B. (1992). "Neural Networks and Fuzzy Systems." Prentice Hall

Mahey, P. (1987). "Programação Não-Linear: Introdução à Teoria e aos Métodos". Editora Campus Ltada.

Mateus, G.R.; Luna, H.P.L. (1186). "Programação Não Linear." . V Escola de Computação- Belo Horizonte, MG-Brasil.

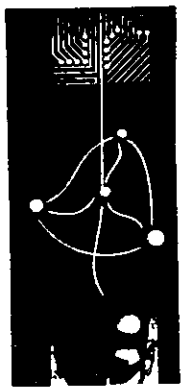
Nordin, K.B.; Novotny, D.W; Zinger, D.S. (1985). "The Influence of Motor Parameter Deviations in Feedforward Fiel Orientation Drives Systems". IEEE Trans. on Industry Aplications, Vol.1A - 21, No 4, pp 1009 - 1015.

Okuyama, T.; Nagase, H.; Kubota, Y.; Horiuchi, H.; Miyazaki, K.; Ibori, S. (1983). "High Performance A.C. Motor Speed Control System Using GTO Converters", Anais do IPEC pp 720-731, Tokyo - Japão.

Saldanha, R.R. (1992). "Optimisation en Eletromagnétism par Application Conjointe des Méthodes de Programmation Non Linéaire et de la Méthode des Eléments Finis.", Tese de Doutorado - De L'Institut National Polytecnnique de Grenoble, França.

Zuben, F.J.V. (1993). "Redes Neurais Aplicadas ao Controle de Máquinas de Indução.", Dissertação de Mestrado, Curso de Pós-Graduação em Engenharia Elétrica da UNICAMP/SP.

Widrow, B.; Lehr, M.A. (1990). "30 Year of Adaptive Neural Networks: Perceptron, Medaline, and Backpropagation.", Proceedings of the IEEE, Vol. 78, No. 9, pp 1415-1442.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

IMPLEMENTAÇÃO EM LABORATÓRIO DE UM ESTIMADOR DE SISTEMAS DINÂMICOS NÃO-LINEARES USANDO REDES NEURAIS ARTIFICIAIS

Sérgio R. J. Oliveira e Edilberto P. Teixeira
Universidade Federal de Uberlândia
Departamento de Engenharia Elétrica
38400 Uberlândia - MG
Tel: (034) 235 2888 R 166
Fax: (034) 236 5099

RESUMO

Este artigo apresenta um método para estimação da dinâmica direta de uma classe de sistemas dinâmicos nãolineares. O método considera que as equações do sistema sejam desconhecidas. Para isso, aplica-se uma rede neural de várias camadas, capaz de estimar um mapeamento que aproxima a dinâmica direta do sistema, dentro de uma determinada faixa de operação. O método foi aplicado na estimação da dinâmica direta do conjunto composto por um motor C.C., acoplado a um gerador de indução e acionado por um retificador controlado de 6 pulsos. Os resultados apresentados foram obtidos da montagem realizada no laboratório de Automação e Robótica do Departamento de Engenharia Elétrica da Universidade Federal de Uberlândia.

1 - INTRODUÇÃO

As redes neurais artificiais (RNA) têm sido aplicadas nas mais diversas áreas, desde o desenvolvimento dos princípios básicos da computação neural por Mc Cullow e Pitts, e do estabelecimento da regra de Hebb [1]. A criação do algoritmo de propagação retroativa [1], para o treinamento de redes neurais de múltiplas camadas (RNMC), tem contribuído para o grande uso desse tipo de rede. As redes neurais de várias camadas se aplicam muito bem para a estimação de mapeamentos não-lineares. Neste sentido, este trabalho trata de uma aplicação das redes neurais de várias camadas, na estimação da dinâmica direta um sistema não-linear implementado em laboratório. Inicialmente, apresentam-se as classes de sistemas não-lineares às quais se aplica o método em questão. Em seguida, é feita uma descrição do método proposto, para estimação do sistema não-linear. Finalmente, são apresentados os resultados da aplicação do método para a estimação da dinâmica direta de um sistema não-linear composto por um motor C.C. acoplado a um gerador de indução e excitado por um retificador controlado de 6 pulsos, montado em laboratório.

2 - SISTEMAS NÃO-LINEARES

Um sistema não-linear, contínuo no tempo, pode ser colocado na forma da seguinte equação:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)) \\ y(t) &= h(x(t)) \end{aligned} \qquad (1)$$

onde x é a variável de estado que pertence a um conjunto aberto **X** de $\mathbf{R}^n$; u é a variável de controle que pertence a um conjunto aberto **U** de $\mathbf{R}^m$; y é a saída do sistema que pertence a um conjunto aberto **Y** de $\mathbf{R}^p$; t é a variável independente representando o tempo contínuo; f é um mapeamento suave de $\mathbf{R}^{m+n}$ para $\mathbf{R}^n$, e h é também um mapeamento suave de $\mathbf{R}^n$ para $\mathbf{R}^p$. Esta é uma classe de sistemas bastante geral porque somente se aplica a restrição de que f e h sejam suaves. O estabelecimento de um procedimento geral para o controle de tais sistemas é uma tarefa bastante complexa. A maioria dos procedimentos seriam altamente dependentes das características de f e h. Além do mais, a abrangência do esquema de controle estaria restrita aos conjuntos **X**, **Y**, e **U**.

Alguns procedimentos de controle consideram sistemas da seguinte classe:

$$\dot{x}(t) = f(x(t)) + \sum_{i=1}^{m} g_i(x(t))\, u_i(t) \quad (2)$$

$$y(t) = h(t)$$

onde x é a variável de estado pertencente a um conjunto aberto **X** de $\mathbf{R}^n$; u é a variável de controle pertencente a um conjunto aberto **U** de $\mathbf{R}^m$; y é saída do sistema pertencente a um conjunto aberto **Y** de $\mathbf{R}^p$ e t é a variável independente representando o tempo contínuo. Neste caso, f e g_i são mapeamento suaves de $\mathbf{R}^n$ para $\mathbf{R}^n$; h é também um mapeamento suave de $\mathbf{R}^n$ para $\mathbf{R}^p$. Esta é uma classe mais restrita de sistemas, visto que, algumas plantas industriais somente podem ser colocadas na forma da equação (1). O método de linearização por realimentação [4] é um exemplo de procedimento que exige que o sistema esteja na forma da equação (2). Problemas importantes, tais como a determinação da controlabilidade [6] e desacoplamento de entrada-saída [7], foram analisados para essas classes de sistemas. A solução do problema de linearização por realimentação para sistemas não-identificados da classe (1) foi proposto em [4]. O problema da identificação foi tratado em [9], usando redes neurais de várias camadas.

No caso específico, deste trabalho, não é possível enquadrar o sistema implementado em laboratório na classe (2), devido à consideração do retificador controlado e da carga constituída pelo gerador de indução. Assim sendo, deve-se enquadrá-lo na classe (1), que é uma classe mais geral de sistemas não-lineares.

3 - O SISTEMA DINÂMICO

O sistema dinâmico estimado é composto por um motor C.C. de 220 volts / 370 watts acoplado a um gerador de indução, também de 220 volts / 370 watts, ligado a uma carga capacitiva de 10 μF. O motor C.C. é alimentado por um retificador trifásico de 6 pulsos a tiristor. A tensão fornecida pelo retificador ao motor é dependente do ângulo de disparo dos tiristores. Este ângulo é fornecido pelo computador em forma binária que é então convertido em um nível de tensão analógica por um conversor digital / analógico de 8 bits. Esta tensão é aplicada ao circuito gerador de disparo que a utilizara como referência para geração de ângulo de disparo. É importante destacar que o retificador trifásico é, por si só, um sistema não-linear já que a relação entre a tensão de saída do retificador e o ângulo de disparo é senoidal. De acordo com o exposto, a tensão aplicada à armadura do motor C.C. é dependente do ângulo de disparo fornecido pelo computador, o que garante um controle total da excitação do motor C.C.. A realimentação, isto é, a leitura da velocidade de rotação do motor é realizada por um medidor ótico de velocidade acoplado ao eixo do motor. Este medidor fornece uma tensão C.C., proporcional a velocidade de rotação do motor. Um conversor analógico / digital de 8 bits converte esta tensão em um número binário que é lido pelo computador. O computador controla as informações sobre o comportamento do sistema, a qualquer instante, através do circuito de realimentação e excitação. Utilizou-se o tempo mínimo de amostragem igual a 130 ms. Este tempo de deve aos atrasos inerentes aos dispositivos de medida.

Na seção seguinte descreve-se a rede neural usada na estimação da dinâmica do conjunto.

4 - TREINAMENTO DA REDE NEURAL

Para a estimação da dinâmica direta do sistema dinâmico, foi utilizada uma rede neural do tipo Perceptron com uma camada

intermediária. A camada de entrada é formada por 10 elementos, a camada intermediária por 350 elementos e a camada de saída por 1 elemento. O algoritmo de treinamento utilizado foi o de Propagação Retroativa [11]. Os seguintes passos são seguidos para o treinamento da rede neural.

1) Apresentam-se, à entrada da rede, as 4 últimas velocidades lidas do motor C.C. e o ângulo de disparo que é também enviado ao retificador. Este ângulo é gerado aleatoriamente pelo computador com distribuição uniforme de probabilidade. Isto garante um aprendizado uniforme dentro da região de operação.

2) Calcula-se a saída da camada intermediária com a seguinte equação:

$$h_k = \frac{1}{1+e^{-\Sigma_i v_{ik} S_i}} \quad (3)$$

onde:

V_{ik} = Matriz de pesos entre a camada de entrada e a intermediária.

S_i = Camada de entrada.

3) Calcula-se a saída da rede com a seguinte equação:

$$u_j = \frac{1}{1+e^{-\Sigma_k w_{kj} h_k}} \quad (4)$$

onde:

W_{kj} = Matriz de pesos entre a camada intermediária e a de saída.

h_k = Saída da camada intermediária.

4) Calcula-se o valor de δ_j a ser utilizado na atualização da matriz de pesos W_{kj}.

$$\delta_j = u_j(1-u_j)\,(t_j-u_j) \quad (5)$$

onde:

u_j = Saída da rede.

t_j = Valor desejado para a saída da rede, que no caso é a velocidade atual de rotação do motor.

5) Calcula-se o valor de $\Delta W_{kj}(n)$ a ser adicionado à matriz de pesos W_{kj}.

$$\Delta w_{kj}(n) = \eta \delta_j h_k + \alpha \Delta w_{kj}(n-1) \quad (6)$$

onde:

η = Taxa de aprendizagem.

δ_j = Erro entre o valor desejado e o obtido na saída da rede.

α = Fator de amortecimento.

$\Delta W_{kj}(n-1)$ = Último valor adicionado à matriz de pesos W_{kj}.

No projeto em questão os valores escolhidos para η e α são:

$\eta = 0.05$

$\alpha = 0.0$

6) Calcula-se o valor de δ_k^* a ser utilizado na atualização da matriz de pesos V_{ik}.

$$\delta_k^* = h_k(1-h_k)\,\Sigma_j \delta_j w_{kj} \quad (7)$$

7) Calcula-se o valor de $\Delta V_{ik}(n)$ a ser adicionado à matriz de pesos V_{ik}.

$$\Delta v_{ik}(n) = \eta \delta_k^* S_i + \alpha \Delta v_{ik}(n-1) \quad (8)$$

onde:

$\Delta V_{ik}(n-1)$ = Último valor adicionado à matriz de pesos V_{ik}.

8) Atualizam-se as matrizes de pesos W_{kj} e V_{ik} com os valores ΔW_{kj} e ΔV_{ik}, respectivamente.

9) Repetem-se os passos de 1 a 8 por aproximadamente 1.000 vezes, e apartir daí a rede é colocada no modo de reconhecimento onde seu desempenho é avaliado.

Na figura 1 tem-se um diagrama de blocos para o treinamento da rede.

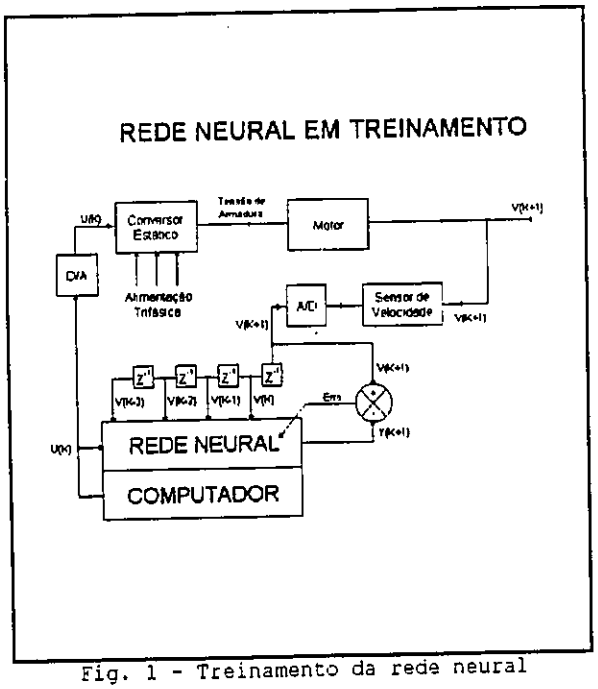


Fig. 1 - Treinamento da rede neural

5 - OPERAÇÃO DA REDE NEURAL

A figura 2 apresenta um diagrama de blocos de todo o sistema, com a rede neural no modo de operação.

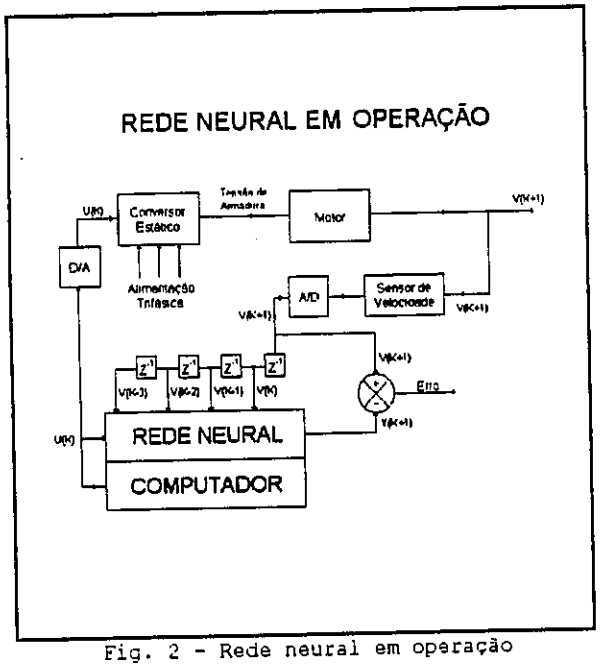


Fig. 2 - Rede neural em operação

Neste modo, a camada de entrada da rede é excitada com o ângulo de disparo atual, que também é fornecido ao retificador, e as 4 últimas velocidades anteriormente lidas do motor. A saída da rede será a próxima velocidade de rotação do motor para o ângulo de disparo atual. Assim, esta velocidade, juntamente com a velocidade atual, lida do motor, são mostradas em um gráfico, onde é feita uma comparação entre as mesmas. Se o resultado ainda não atingiu a precisão desejada a rede é novamente colocada no modo de treinamento.

6 - RESULTADOS

Após decorridas 1000 iterações, o processo de treinamento da rede neural foi iterrompido para avaliar-se o desempenho da mesma. Para isto, a rede foi colocada no modo de operação e apartir dai, foram armazenadas as velocidades apresentadas pela rede neural e pelo motor C.C., para as mesmas excitações. A figura 3 mostra o gráfico com as referidas velocidades.

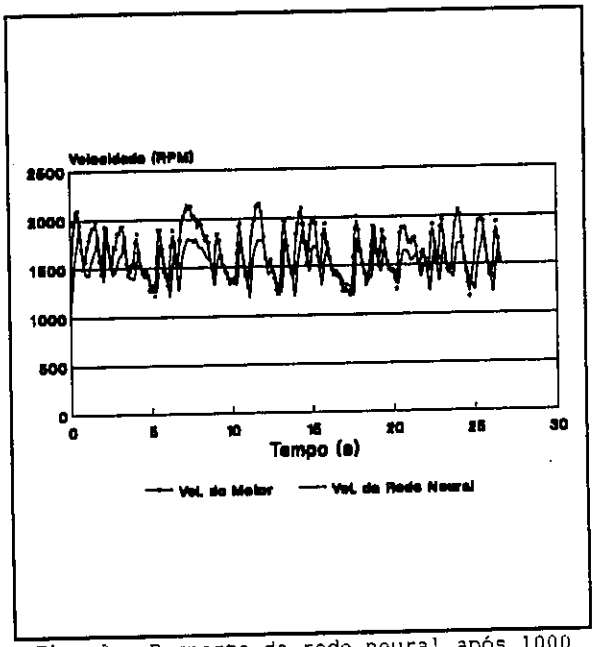


Fig. 3 - Resposta da rede neural após 1000 iterações

Observa-se que, com apenas 1000 sessões de treinamento a rede neural não consegue acompanhar a velocidade apresentada pelo motor C.C., cometendo erros absurdos. Face a isto, a rede foi colocada no modo de treinamento até 7000 iterações. Neste ponto, a rede foi novamente colocada no modo de operação para avaliar-se o seu desempenho. A figura 4 mostra a resposta da rede neural após esta segunda seção de treinamento. Na figura, observa-se que a resposta de velocidade da rede já se aproxima muito da velocidade do motor

C.C.. O erro cometido pela rede, neste caso, é muito menor que o observado após 1000 iterações.

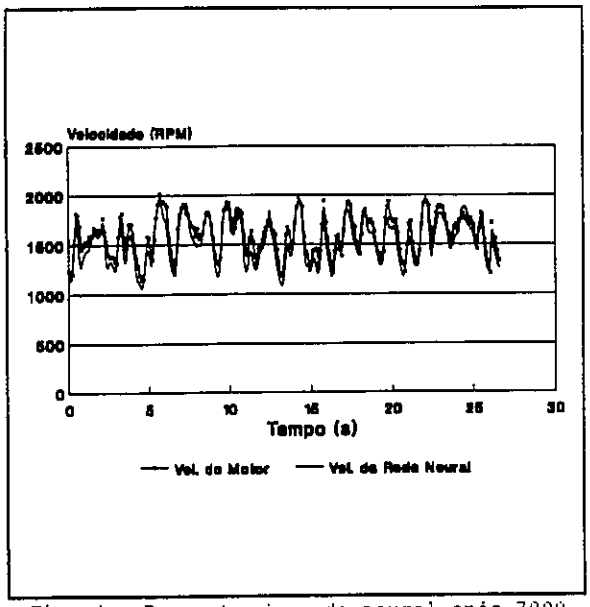


Fig. 4 - Resposta da rede neural após 7000 iterações

Como houve uma grande melhoria na assimilação das características do sistema não-linear, com esta segunda seção de treinamento, a rede foi novamente colocada no modo de treinamento. Desta vez, após 18000 e 25000 iterações, o seu desempenho foi avaliado, obtendo-se os mesmos resultados, conforme mostrado na figura 5.

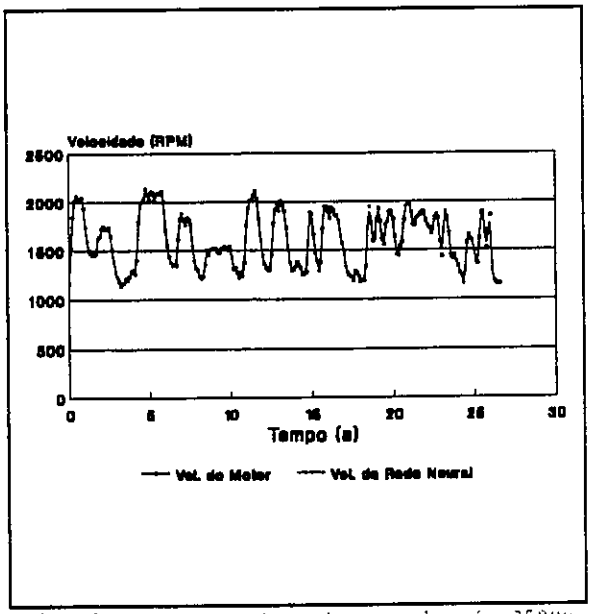


Fig. 5 - Resposta da rede neural após 25000 iterações.

Neste ponto, a rede neural praticamente já assimilou todas as características do sistema a ela conectado e o erro manteve-se a um valor máximo de 30 RPM.

7 - CONCLUSÃO

Os resultados obtidos experimentalmente em laboratório mostram que a rede neural aprendeu a dinâmica direta do sistema com muita exatidão. Deve-se salientar que os testes foram feitos em laboratório utilizando-se equipamentos comuns no mercado, o que viabiliza a sua aplicação prática. Embora tenha-se utilizado um computador IBM 486 para treinamento da rede, pretende-se em um futuro próximo testar um circuito dedicado que está sendo desenvolvido em nosso laboratório. Está sendo também implementado o processo de linearização por realimentação desenvolvido em [4] e que será apresentado em uma próxima publicação.

8 - REFERÊNCIAS

[1] Rumelhart, D. McClelland, J., "Parallel Distributed Processing", Vol. 1, MIT Press, Cambridge, MA, 4th Edition, 1987.

[2] Loparo K. & Teixeira E., "A new approach For Adaptive Control of Nonlinear Systems Using Neural Networks", IEEE Int. Conf. Sys. Man Cyb., Los Angeles, 1990.

[3] Teixeira E, Loparo K, Gomide F, "Design Multi-layer Neural Networks for Accurate Identification of Nonlinear Mappings", American Cont C., Boston, 1991.

[4] Teixeira E., Loparo K., Gomide F., "Feedback Linearization of Dynamic Nonlinear Systems Using Neural Networks", 9º Cong. Bras. Autom., Vitória, 1992.

[5] Nie J. & Linkens D., "Neural Network-based approximate reasoning: principles and implementation", Int. Journal of Control, vol. 56, nº 2, pp. 399-413, 1992.

[6] Stefani G. "On the Local Controllability of a Scalar-Input Control System", Theory and Application of Nonlinear Control Systems, Elsevier Science, 1986.

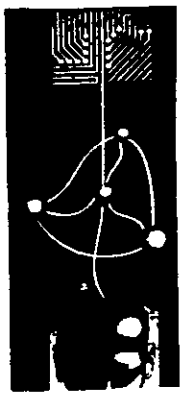
[7] Singh R. & Rugh W. "Decoupling Class of Nonlinear Systems by State Variable Feedback", Transactions ASME J.D.Syst.M.Cont, V.21, pp. 651-654, 1975.

[8] Nguyen D. & Widrow B. "Neural Networks for Self-Learning Control Systems", IEEE Control Magazine Systems, April 1990.

[9] Narendra,K., Parthasarathy, K. "Identification and Control of Dynamical Systems Using Neural Networks", IEEE Transactions on Neural Networks, Vol.1, #1, March 1990.

[10] Oliveira, S. R. J. & Teixeira, E. P. "Identificação da Dinâmica Inversa de Sistemas Não-Lineares Através de Redes Neurais Artificiais", I SBAI - UNESP Rio Claro, Setembro 1993.

[11] David, J. B. "Experiments on Neural Net Recognition of Spoken and Written Text", IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 36, #7, July 1988.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajuba
Itajuba, 24 a 27 de outubro de 1994

CONTROLE AUTOMÁTICO DE SISTEMAS LINEARES EMPREGANDO MODELOS DE REDES NEURONAIS

Roberto Celio Limão de Oliveira

Departamento de Engenharia Eletrica
Centro Tecnologico
Universidade Federal do Para
Caixa Postal 6025, 66.075.900
Belem, PA

Takashi Yoneyama

Divisão de Engenharia Eletronica
Instituto Tecnologico de Aeronautica
CTA - ITA - IEEE
2.225-000
São José dos Campos, SP

RESUMO

Este trabalho descreve a utilização de um modelo de rede neuronal para ser utilizado no controle de um sistema dinâmico. A rede neuronal é treinada como um controlador *feedforward*. Utiliza-se uma técnica simples de aquisição de dados na entrada do modelo neuronal, através de sinais atrasados no tempo. Os pesos representando a intensidade de conexão das sinapses são modificados via algoritmo de aprendizado padrão do tipo *backward error propagation*. Após a fase de treinamento, o controlador neuronal apresenta rapida velocidade de resposta, a qual depende somente do numero de niveis de neurônios e do tempo de propagação do sinal atraves dos mesmos. Os resultados obtidos com o controlador neuronal são comparados com os conseguidos com um controlador PID.

ABSTRACT

This work describes the implementation of a neural network model to be used in the control a dynamical system. The neural network is trained to be used as a feedforward controller. A simple data acquisition technique is applied at the neural network input. A delayers signals bank is used. The weights representing the synaptic connection strenghtes are changed through an backward error propagation learning algorithm. After the training phase, the neural network controller presents a very high response speed with depends only on the number of layers and the time for signal propagation through the neurons. The results obtained with the neural controller are compared with the results of a PID controller.

1. INTRODUÇÃO

Ações de controle são atos naturais aos homens e frequentemente embutido nas maquinas. Existem diferenças significativas entre o controle executado por humanos e o feito por equipamentos. Pessoas fazem o uso de um numero grande de informações no planejamento e execução de uma ação de controle quando comparado com controladores indusriais. A razão desta diferença não esta nas limitações da disponibilidade de entradas sensoras, mas sim na capacidade dos controladores de processar com sucesso esta quantidade de informações.

Uma outra distinção entre o homem e a máquina é a maneira coletiva de processar informações que o cerebro apresenta, dando ao mesmo a habilidade de responder rapidamente a complexas informacões sensoras, sendo que os mais sofisticados algoritmos de controle tem serias limitações de tempo quando

aplicados em tempo real. Uma terceira distinção e a mais importante, é que a ação humana baseia-se no aprendizado, enquanto a operação de controladores artificiais requerem um algoritmo escrito a priori. Devido a estes motivos e com os novos conhecimentos da neurobiologia sobre o funcionamento do cérebro é que surge a idéia de utilizar-se modelos de redes neuronais em sistemas de controle. O intrínsico paralelismo no processamento das informações apresentado por estes modelos, a rápida velocidade de resposta com o qual este processamento é executado e a característica de aprendizado tem despertado o interesse da engenharia de controle.

Um controlador que usa uma arquitetura de rede neuronal exibe três características importantes: utilização de *várias informações sensoras, capacidade* de processamento coletivo e adaptação [5]. Neste artigo, utilizaremos um modelo de rede neuronal do tipo *perceptron* multi-camadas como um controlador *feedforward*. E apresentado um exemplo do seu uso para o controle de um sistema dinâmico linear e os resultados também são comparados com o uso do controlador PID.

2. MODELO NEURONAL TIPO PERCEPTRON

Modelos de redes neuronais tem sido estudados na expectativa de se conseguir a mesma performance que os humanos em algumas tarefas específicas, tais como : reconhecimento de voz e de imagem, controle da atividade motora, problemas de classificação e decisão [6].

Um modelo de rede neuronal é construido a partir de uma estrutura formada por elementos processadores (neurônios), simples e interconectados. Estes elementos processadores operam em paralelo e estão organizados segundo padrões tirados do atual conhecimento biológico [3].

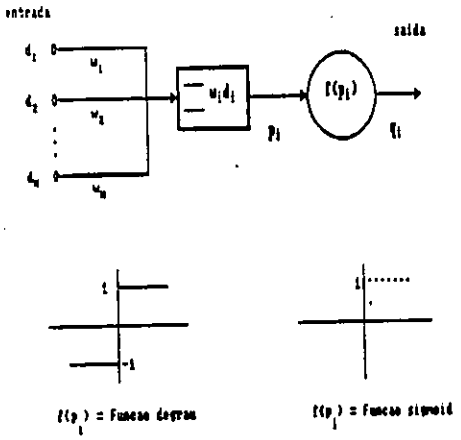


Figura 2.1 - Elemento Processador

Para que o modelo neuronal apresente um bom desempenho, utiliza-se uma densa interconexão dos elementos processadores. Estes são conectados via pesos que são adaptados durante o uso, visando a melhora de um critério de desempenho pré-estabelecido. O modelo neuronal *perceptron* multicamada são redes *feedforward* com um ou mais níveis de elementos processadores entre os elementos processadores de entrada e de saída, chamados de níveis escondidos.

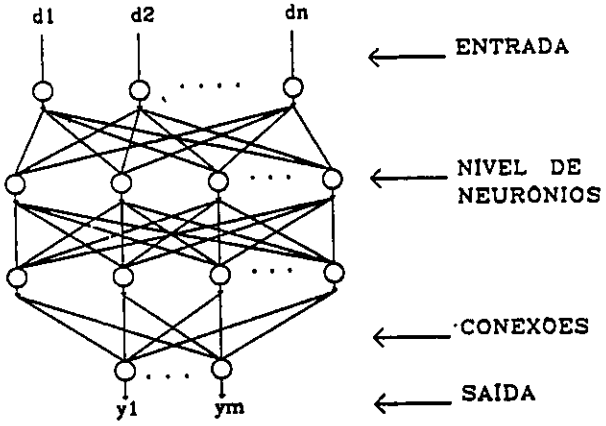


Figura 2.2 - Modelo Neuronal Multicamadas

O procedimento de aprendizado é conseguido por meio do algoritmo *backward error propagation*. Este

iterativo do gradiente que minimiza uma função custo quadrática J, que é função do erro entre a saída desejada para o modelo neuronal e saída atual do mesmo [6].

$$J = \frac{1}{2}(u_d - u)^2 \qquad (1)$$

3. CONTROLE E APRENDIZADO EMPREGANDO MODELOS DE REDES NEURONAIS

Um sistema onde o modelo neuronal é empregado de modo a funcionar como um controlador *feedforward* é mostrado na figura (3.1), observando-se a saída da planta a ser controlada (y), o padrão de saída desejado (yd), o laço de realimentação (e), o sinal realimentado (uf), a saída do modelo neuronal (ui) e o sinal de controle (ut). O sinal realimentado uf(t) é utilizado como o sinal de erro (ud - u) no algoritmo de aprendizado.

são suficientes para que a rede neuronal encontre, via aprendizado, os valores dos pesos necessários a ação de controle desejada, onde o número de sinais atrasados no tempo é determinado pela ordem da planta. A utilização de um *banco de atrasos* na entrada do modelo neuronal determina uma técnica simples de aquisição de sinais, de modo a fornecer ao controlador neuronal as informações relativas a dinâmica desejada para o processo controlado, facilitando o uso deste tipo de controlador em operações de tempo real.

O controle e o aprendizado são feitos simultaneamente, observando-se que a medida que a rede vai aproximando-se da ação de controle desejada, o sinal realimentado deixa de ter influência no controle ficando apenas o sinal de saída do modelo neuronal a comandar a planta.

4. EXEMPLO APLICATIVO

Utiliza-se como planta a ser

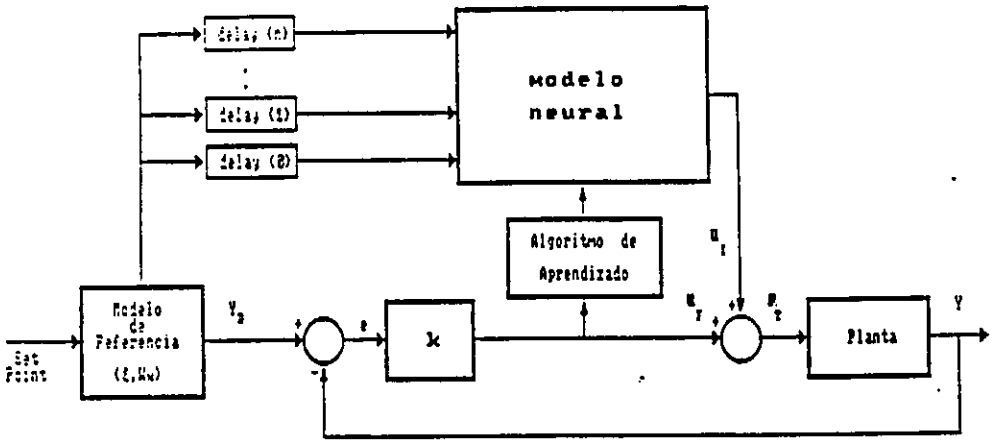


Figura 3.1 - Arquitetura de Controle e Aprendizado

Esta estrutura de controle assume o não conhecimento sobre a dinâmica da planta, exceto pela ordem da mesma. Sendo que na entrada do modelo neuronal temos apenas um conjunto de sinais atrasados no tempo. O uso dos sinais controlada, um sistema servo-posicionador, sendo que o problema de controle incorpora tanto o caso de regulador como o caso de seguimento de trajetória. Nas simulações é utilizada a

$$G(s) = \frac{600}{s^3 + 50s^2 + 600s} \qquad (2)$$

A rede neuronal utilizada é composta de três níveis, sendo que há quatro elementos processadores no nível de entrada, cinco elementos processadores no nível escondido e um elemento processador no nível de saída. O nível de entrada recebe os quatro sinais de entrada yd(t), yd(t-1), yd(t-2) e yd(t-3) onde observa-se a informação da ordem da planta n=3. O último nível tem como saída do único elemento processador o sinal de controle para a planta, sendo que a função ativação deste elemento processador é uma função linear com saturação em valores iguais a ± 10, indicando limitações físicas no sinal de controle. Nos outros níveis, todas as funções ativações são do tipo sigmoide.

Utiliza-se uma constante de aceleração no algoritmo de aprendizado de 0.025 e pesos iniciais aleatórios com distribuição normal de média zero e variância 0.001. A curva desejada, que segue o modelo de referência de segunda ordem discreto [1], tem como parâmetros o coeficiente de amortecimento igual a 0.8, frequência natural igual a 1.7 rad/seg e período de amostragem de 0.1 segundos, permitindo um tempo de subida igual a 2.4 segundos e o percentual de sobresinal máximo de 0.01%. O controlador de realimentação que tem como entrada e = yd-y e gera na saída uf = k*e, tem como parâmetro k=1.

Na figura (4.1) está representado apenas o laço de realimentação, sem o sinal de saída do modelo neuronal. Aqui, temos a ação de um controlador proporcional de ganho unitário, sendo que esta ação de controle fornecerá o sinal de treinamento para o modelo neuronal.

aprendizado, na encontrar no espaço de pesos a melhor configuração para a tarefa pré-fixada, podendo-se prever uma certa demora na excursão dos pesos por todo o espaço de estados possíveis.

Verifica-se que, com o aumento do número de iterações, onde cada iteração representa a apresentação de toda a curva desejada e o respectivo ajuste de pesos, na fase de treinamento, a influência do sinal de controle realimentado (uf(t)) diminui gradativamente, sendo mais influente o sinal de saída do modelo neuronal. A rede após o treinamento guarda na forma dos pesos as características desejadas para o controlador, sendo que estes pesos estão ajustados para o padrão de saída desejado pré-fixado. Com estes valores de pesos, o modelo neuronal pode adaptar-se a uma nova situação, representando um conhecimento adquirido para o qual não tenha sido explicitamente treinado. Esta característica da rede neuronal pode ser encarada como um processo de interpolação [8].

A *partir de uma nova trajetória* desejada, verificou-se o comportamento de saída da planta com o modelo neuronal estando com os pesos fixados pelo treinamento inicial e o algoritmo de aprendizado fora de operação. Para este novo padrão desejado tem-se como parâmetros do modelo de referência um coeficiente de amortecimento igual a 0.6 e frequência natural de 1.5 rad/seg, acarretando um tempo de subida no valor de 1.8 seg e um percentual de sobresinal máximo igual a 0.09%.

Como pode ser observado na figura (4.4), o modelo neuronal adapta-se a mudanças no padrão de saída desejado, caracterizando o processo de interpolação para pontos não treinados.

Para verificar o comportamento do modelo neuronal face a presença de ruídos, a partir dos pesos treinados inicialmente (pesos responsáveis pelo comportamento apresentado na figura

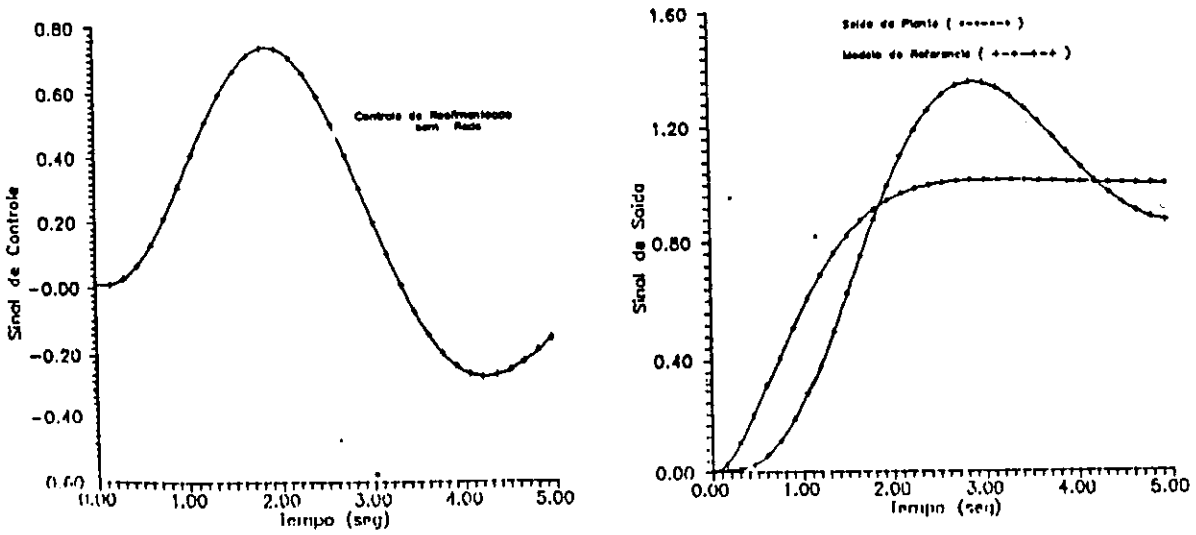


Figura 4.1 - Sinal de Controle e Saída da Planta antes do treinamento

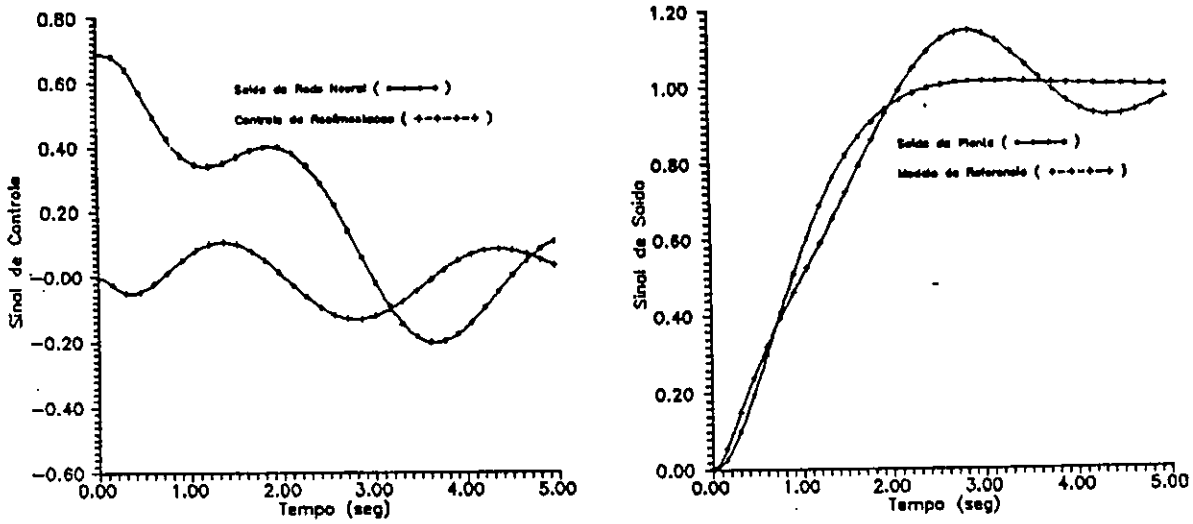


Figura 4.2 - Sinal de Controle e Saída da Planta após 5000 iterações

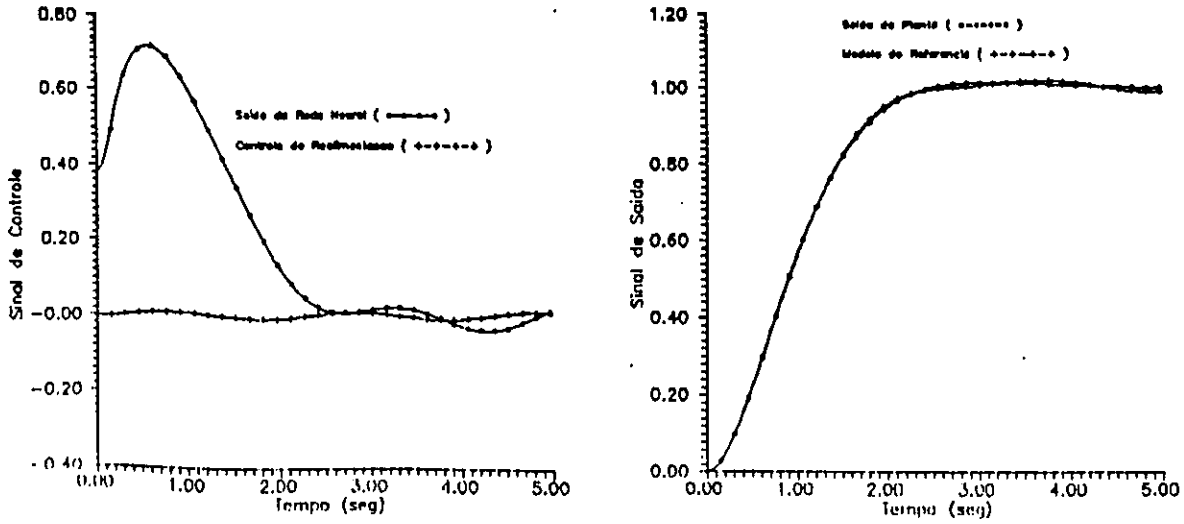


Figura 4.3 - Sinal de Controle e Saída da Planta após treinamento (10000 iterações)

sensor, estando novamente o algoritmo de aprendizado desligado. A performance do controlador neuronal e mostrada na figura (4.5), onde observa-se o bom desempenho do mesmo.

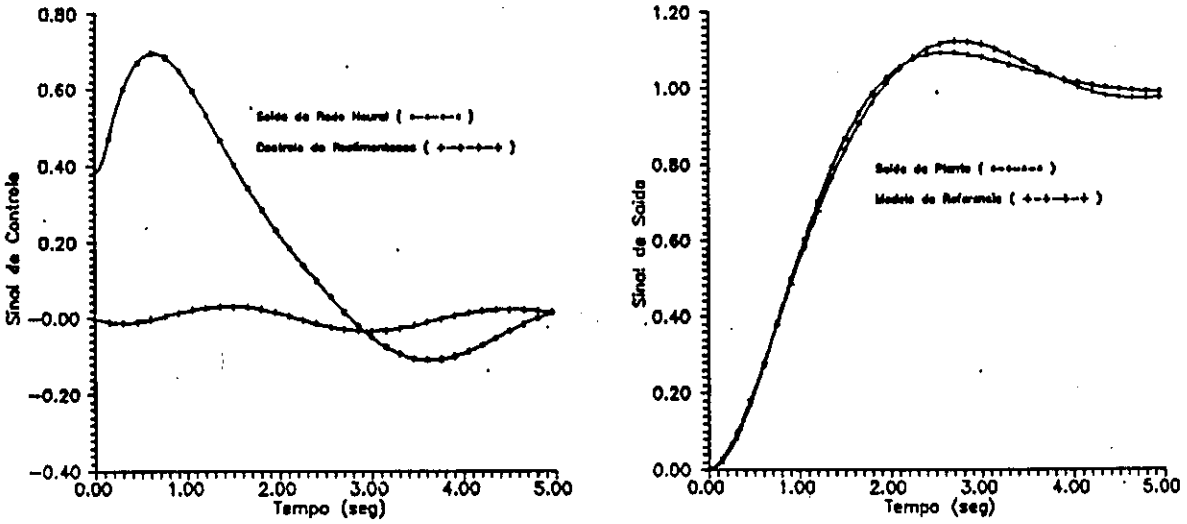


Figura 4.4 - Sinal de Controle e Saida da Planta após mudanças no padrão desejado

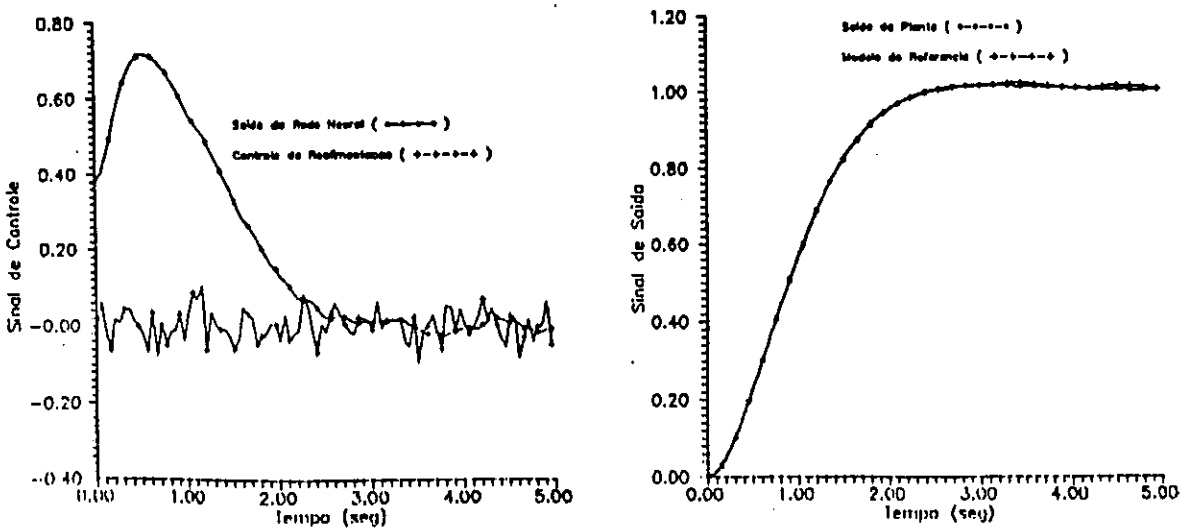


Figura 4.5 - Sinal de Controle e Saida da Planta com ruído no sensor

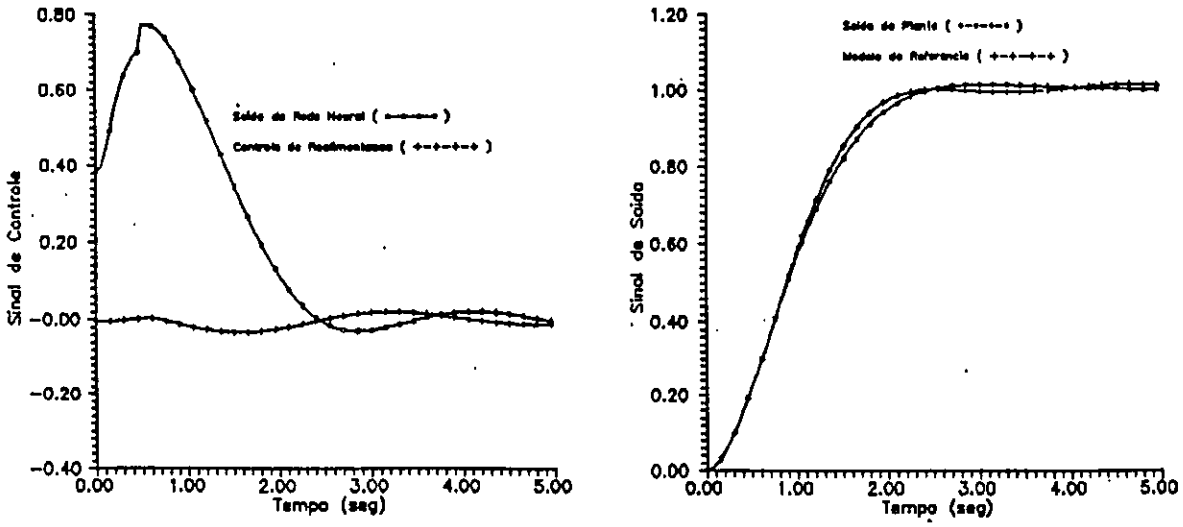


Figura 4.6 - Sinal de Controle e Saida da Planta com variação na carga de 11%

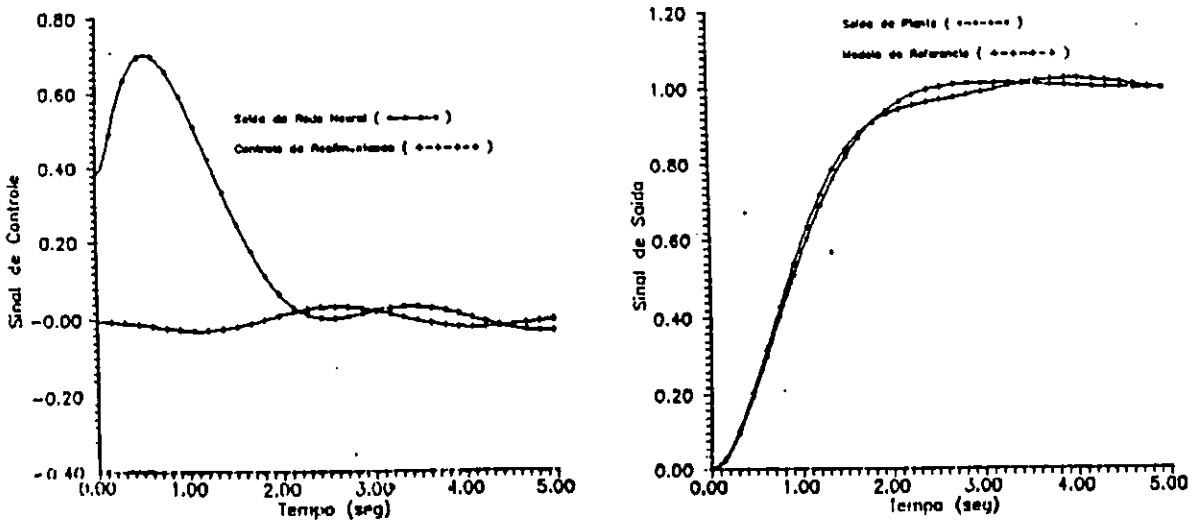


Figura 4.7 - Sinal de Controle e Saída da Planta após modificação no valor do peso

Para avaliar o controlador tendo em vista modificações nos parâmetros da planta, modificou-se a carga no eixo do servo-posicionador em mais 11%, continuando o modelo neuronal a ter os pesos responsáveis pelo controle da figura (4.3) e o aprendizado não ativo. Como pode-se observar na figura (4.6), o controlador neuronal apresenta boa performance, mesmo na presença de mudanças nos parâmetros da planta.

Observando-se agora o comportamento do modelo neuronal quando ocorre alguma falha no mesmo, modificou-se externamente o peso que liga o terceiro neurônio do primeiro nível com o terceiro neurônio do segundo nível, à metade do seu valor original, onde este é dado pelo treinamento da figura (4.3). O resultado desta falha é apresentado na figura (4.7), com o aprendizado inativo, observando-se um desempenho razoável para o controlador neuronal, com alguma degradação de performance, determinada principalmente pela importância do peso no conjunto de pesos disponíveis. Sendo que está importância diminui a medida que o conjunto de pesos aumenta.

Nos quatro casos de modificações no ambiente para o qual o modelo neuronal foi previamente treinado, podemos considerar como pontos de treinos não distantes do ponto inicial. Para os casos em que as mudanças ocorridas são bem acentuadas, e os pesos inicialmente treinados não são capazes de gerar o apropriado sinal de coutrole, recomenda-se a utilização de uma realimentação hormonal no controlador neuronal.

Para comparar o desempenho do

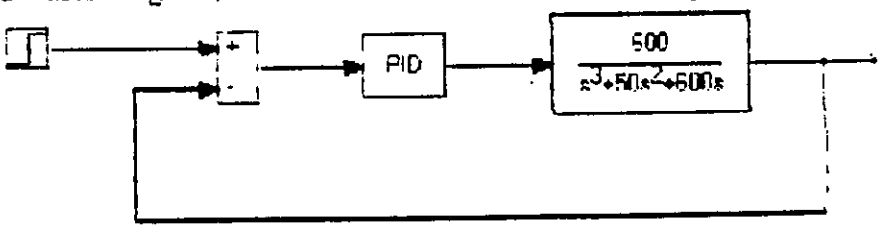


Figura 4.8 - Planta com controlador PID

controlador neuronal com um controlador clássico do tipo PID, montou-se o sistema mostrado na figura (4.8), sendo necessário sintonizar os ganho do PID de tal forma que o mesmo tivesse um desempenho que

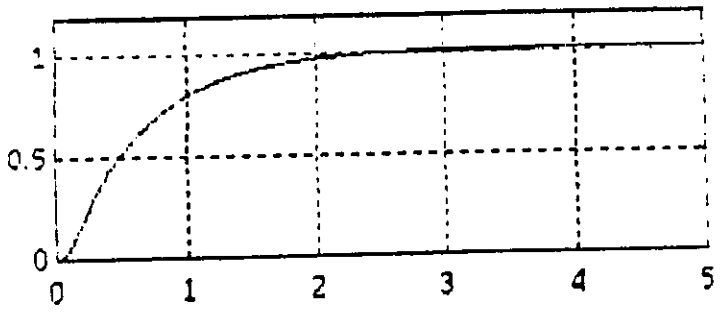


Figura 4.9 - Saída da Planta com o controlador PID

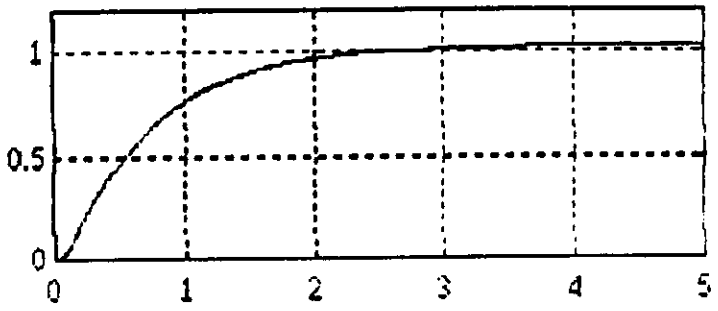


Figura 4.10 - Saída da Planta com variações paramétricas e controlador PID

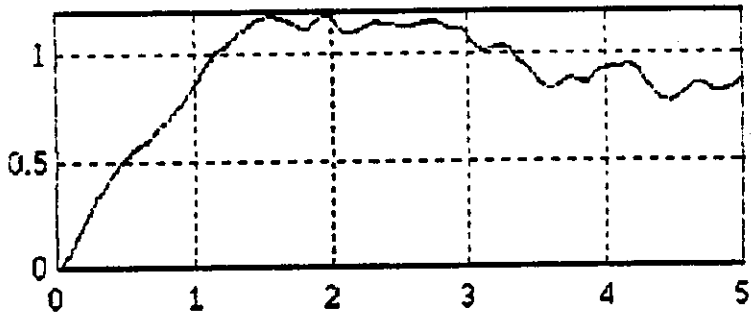


Figura 4.11 - Saída da Planta com ruído no sensor e controlador PID

tornasse a saída da planta próxima ao modelo de referência dado pelo coeficiente de amortecimento igual a 0.8 e frequência natural igual a 1.7 rad/seg. A partir destes ganhos do controlador PID, visto na figura (4.9), verifica-se o comportamento do mesmo frente a problemas de ruído, conforme encontrado na figura (4.5), e a variações de parâmetros da planta, como visto na figura (4.6). Os resultados para o controle através do controlador PID são mostrados nas figuras (4.10) e (4.11), respectivamente para os distúrbios de ruído no sensor e variação paramétrica.

Observando-se estes resultados nota-se o melhor desempenho do controlador neuronal para casos em que haja presença de ruídos, onde a performance do PID é extremamente degradada

E para a situação de variação paramétrica, o comportamento do controlador neuronal apresenta melhores resultados do que o controlador PID, sendo a diferença de performance menos significativa que o apresentado para o ruído.

Nos casos de mudança no padrão de saída desejado e falha nos ganhos do controlador, não foi testado o PID, porque nestes dois casos o trabalho resume-se apenas em nova sintonia dos ganhos do controlador.

5. CONCLUSÕES

A partir das simulações apresentadas, o modelo neuronal *feedforward*, apresenta um bom desempenho. Necessitando para o seu aprendizado apenas a ordem do sistema a ser controlado, utilizado na forma de yd(t), yd(t-1), ..., yd(t-n), onde n é a ordem do sistema a ser controlado.

Na arquitetura apresentada utiliza-se o sinal de realimentação para resolver o problema da necessidade de se conhecer a priori o sinal de controle da planta sem o qual não consegue-se aplicar o algoritmo de aprendizado.

Nota-se que, apesar do numero de iterações necessárias para concretizar-se o treinamento ser grande, com o pequeno número de elementos processadores utilizados no modelo neuronal, o tempo gasto na sessão de treinamento foi de três horas, com o algoritmo sendo executado em uma máquina 386 da HP.

Com base nas figuras (4.4), (4.5) e (4.6), pode-se afirmar que o modelo neuronal utilizado como controlador *feedforward* na arquitetura da figura (3.1), tem bom desempenho quando na presença de ruídos, variacões paramétricas e mudanças no padrão desejado. O conhecimento adquirido no treinamento fica armazenado nos valores dos pesos de uma forma distribuida, tornando o controlador menos sensível a falhas, com esta sensibilidade diminuindo a medida que aumenta-se o número de elementos processadores no modelo.

Relacionando o desempenho do controlador neuronal *feedforward* com um controlador PID, verifica-se que o mesmo apresenta um melhor desempenho, sendo que o PID tem sua performance degradada quando ocorre mudanças no ambiente, tipo variações paramétricas e ruídos.

6. REFERÊNCIAS BIBLIOGRÁFICAS

[1] - Astrom, K. J. & Wittenmark, B. ,"Computer-Controller Systems - Theory and Design", Prentice-Hall Inc., 1990.

[2] - Hoskins, D. A. & Vagners, J., "A Neural Network Based Explicit Model Reference Adaptive Controller", Procedings of the 29th Conference on Decision an Control, Honolulu, Hawaii, December, 1990, pp 1725-1729.

[3] - Lippmann, R. P., "An Introduction to Computing With Neural Nets", IEEE ASSP Magazine, Vol. 3, No 4, April, 1987, pp 4-22.

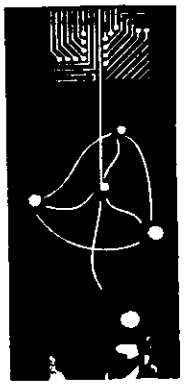
[4] - De Oliveira, R. C. L. & Nascimento Jr., C. L. & Yoneyama, T., "A Fault Tolerant Controller Based on Neural Nets", International Conference Control 91, Edinburgh, March, 1991.

[5] - Psaltis, D. & Sideris, A. & Yamamura, A. A., "A Multi-Layeded Neural Netwok Controller", IEEE Control Systems Magazine, April, 1988, pp 17-21.

[6] - Rumelhart, D. E. & Hinton, G. E. & Williams, R. J., "Parallel Distributed Processing", MIT Press, Vol. 1 and 2, 1986.

[7] - Kawato, M. & Uno, Y. & Isobe, M. & Suzuki, R., "Hierarchical Neural Network Model For Voluntary Movement With Application to Robotics", IEEE Control Systems Magazine, April, 1988, pp 8-15.

[8] - Filho, P. R. B. & De Oliveira, R. P. & Albuquerque, A. B. & De Oliveira, R. C. L. & Garcez, J. N., "Um Estabilizador PID Inteligente para Sistemas de Potência Usando Redes Neuronais Artificiais", a *ser publicado nos Anais do 10° Congresso* Brasileiro de Automática.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

ALGORITMO RÁPIDO DE TREINAMENTO DE REDES NEURAIS PARA SISTEMAS DE CONTROLE EM TEMPO REAL

L.E. Borges da Silva
A.P. Alves da Silva
G. Lambert-Torres
E. R. Ribeiro

Grupo de Inteligência Artificial
Escola Federal de Engenharia de Itajubá

Abstract - The long time generally required for training ANN has been a critical problem for the utilization of this technology in real-time. The generalized delta rule with backward error propagation (backpropagation) has been the most used training algorithm for ANN in control applications. However, despite many attempts to improve the performance of this learning algorithm, there is still no efficient and reliable method to train a multilayer perceptron. The main problem is high nonlinearity of the error function. This paper presents an alternative method for load information into a multilayer perseptron. The idea is to use successive quadratic approximations for the error function, until getting into the proximity of the global minimum. This technique was applied to Control System in order to obtain an adaptive behaviour.

1- Introdução

O uso de Redes Neurais em sistemas de controle, tem se mostrado como uma alternativa muito interessante. Porém, problemas ainda existentes impedem sua utilização no controle adaptativo em tempo real, sendo que, o tempo de treinamento da rede é a restrição mais significativa. A técnica de treinamento mais popular, no momento, a retro-propagação do erro, embora muito simples é uma técnica que consome muito tempo.

Este trabalho apresenta um esquema de controle adaptativo, que usa Redes Neurais para identificação e controle de um acionamento de motor de corrente contínua alimentado por um conversor tiristorizado.

O algoritmo de adaptação e a estratégia de controle são apresentadas. A metodologia proposta foi simulada e os resultados são mostrados. Para alguns problemas detectados, como por exemplo o tempo de convergência, mais precisamente o número de iterações antes do algoritmo atingir um certo valor mínimo de erro, é proposta uma possível solução. Um novo método de carregamento da informação em um Perceptron Multicamadas é apresentado. A idéia é usar aproximações quadráticas sucessivas para a função erro, até que a proximidade de um mínimo global possa ser atingido.

Esta técnica permite o uso do algoritmo "Mínimos Quadrados Recursivos" para computar o mapeamento associativo ótimo. O algoritmo proposto de treinamento para perceptrons multicamadas resolve os dois principais problemas: tempo longo de treinamento e algoritmos complexos de treinamento.

2- Sistemas de Controle Adaptativo

Em muitas aplicações de controle se torna muito difícil representar a dinâmica do processo em equações matemáticas precisas. Processos reais sempre incluem no modelo incertezas tais como variações de parâmetros ou não linearidades, que não são conhecidas no momento do projeto dos controladores.

O controle adaptativo mostrado na figura.1, foi projetado de forma a obter um alto desempenho, independente da variação das características do processo. O problema do

controle é ajustar dinamicamente os parâmetros do controlador de tal forma que a saída da planta, c(k), siga o sinal de referência, r(k). Para isto, a identificação do modelo é utilizada pelo algoritmo adaptativo para o treinamento do controlador.

Muitos são os problemas associados com os controles adaptativos tradicionais; por exemplo, a não existência de uma metodologia aplicável a uma grande quantidade de problemas diferentes.

Neste trabalho apresentamos uma alternativa ao controle adaptativo tradicional para um acionamento de motor de corrente continua.

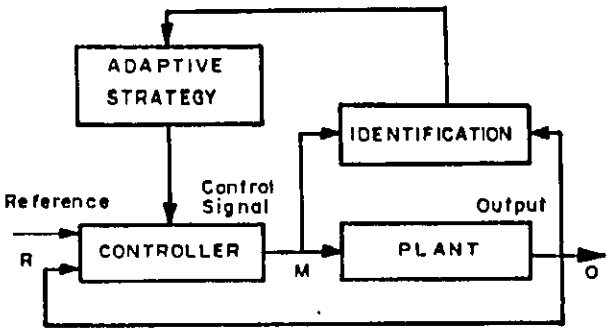


Fig.1 - Arquitetura Tradicional de Controle Adaptativo

3- Redes Neurais em Sistemas de Controle

O elemento básico de processamento de uma rede, o Neurônio, é representado pela equação:

$$h_{out,i} = f\left(\sum_{j=1}^{m} h_{in,j} \cdot w_{ji} - \theta_i\right)$$

O parâmetro θ_i serve como limite ou polarização, $h_{in,j}$ é o conjunto das variáveis de entrada e w_{ji} é o conjunto de pesos das interconecções partindo do elemento **j** para o elemento **i**.

A rede é definida pelo número e o modo como os neurônios são interconectados, pela função de ativação, e o algoritmo de treinamento.

Neste artigo, será considerado somente redes com encaminhamento para frente, também chamadas redes multicamadas (fig.2), devido sua capacidade de aprender características de sistemas através do "mapeamento" não-linear. Esta arquitetura de rede é a mais utilizada para aplicações em controle

A entrada dos neurônios monitoram os sinais externos que são convertidos em saídas, de acordo com os pesos e funções de ativação apropriadas. É sabido que as camadas internas da rede são responsáveis pelo mapeamento não-linear entre sinais de entrada e saída e supressão de padrões de ruídos que porventura possam existir.

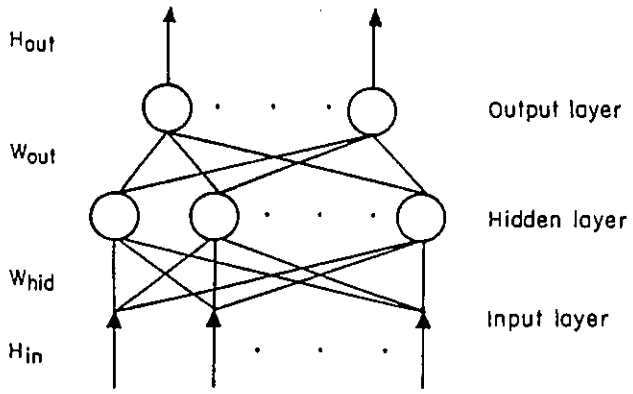


Fig.2 - Rede Neural Multicamadas

A entrada dos neurônios monitoram os sinais externos que são convertidos em saídas, de acordo com os pesos e funções de ativação apropriadas. É sabido que as camadas internas da rede são responsáveis pelo mapeamento não-linear entre sinais de entrada e saída e supressão de padrões de ruídos que porventura possam existir.

Durante o treinamento da rede, os pesos das conecções são ajustados em função da comparação entre o valor real saído da rede e o valor desejado, até que o erro caia dentro de limites especificados.

O método de treinamento mais utilizado, atualmente, é o algoritmo de retro-propagação do erro [1].

4- Identificação e Controle Adaptativo com Redes Neurais

O sistema proposto compreende três partes. A primeira é o treinamento da rede neural para que esta represente corretamente a resposta do sistema. A segunda usa do modelo do sistema identificado, pela rede, para o procedimento de ajuste do controlador. A terceira, quando o sistema estiver funcionando em operação normal, para garantir um bom desempenho, a rede é retreinada continuamente.

A especificação da estrutura da rede (número de entradas, número de saídas, número de camadas internas e número de neurônios), o padrão de treinamento e a escolha dos parâmetros do algoritmo de treinamento, são os principais problemas do projeto de um sistema de controle com redes neurais. Os parâmetros de treinamento possuem um efeito decisivo no desempenho do sistema.

A - Identificação do Processo

O esquema de identificação, mostrado na fig.3, ilustra a posição, em paralelo, da rede com o processo. A cada iteração o padrão de entrada é montado pelo sinal de referência e pelos valores anteriores das amostras da saída do processo. Assim, o problema de identificação consiste no ajuste adequado dos pesos da rede de forma a minimizar o erro E_i (fig.3), isto é, a diferença entre o valor correto do sinal c(k) e da saída da rede c^(k) [4].

Durante o treinamento da rede de identificação, a intervalos regulares, o erro global é calculado e comparado com o valor mínimo desejado que irá determinar o grau de aprendizado da rede.

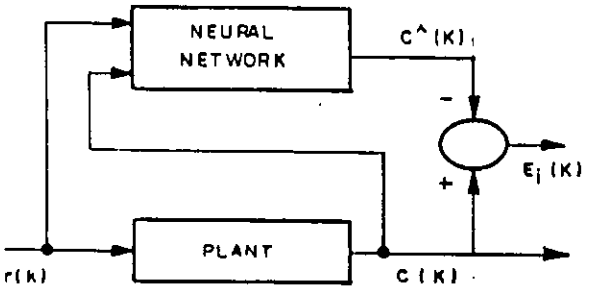


Fig.3 - Esquema de Identificação usando Redes Neurais

B - Implementação do Controle

O objetivo do controle é gerar um sinal m(k), que conduza o processo a um certo ponto de operação desejado. O treinamento do controlador, ilustrado na fig.5, necessita da informação sobre o erro na saída do controlador neural a cada instante. Porém, somente o erro final E_c, entre a saída do processo e a saída desejada, para uma dada referência, está disponível, deste modo, usamos a rede neural como modelo do processo, retro-propagando o erro E_c através da rede de identificação para conseguir um erro equivalente para o controlador.

O ajuste do controlador é similar ao procedimento de identificação, mas agora, o erro global, dado por E_c tem que atingir o valor mínimo desejado. De modo análogo, o sinal de referência tem que seguir as mesmas condições do procedimento de identificação.

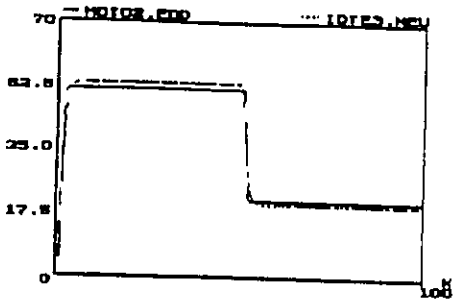


Fig.4 - Resultado da Identificação

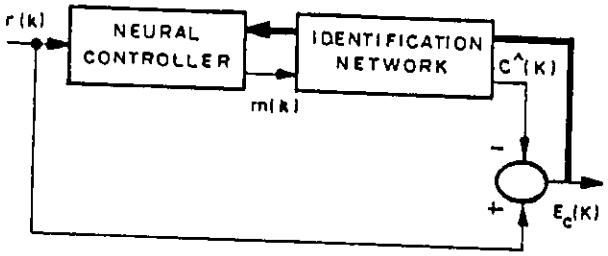


Fig.5 - Esquema de Treinamento do Controlador Neural

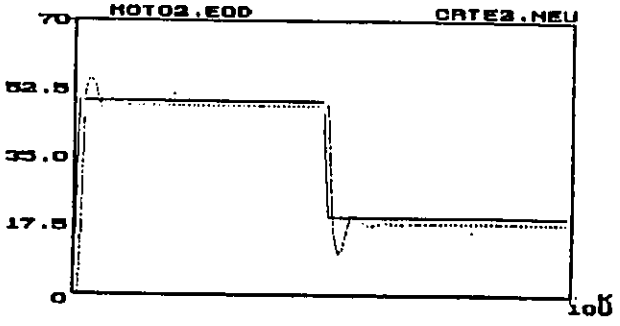


Fig.6 - Resultado do treinamento do Controlador

5- Algoritmo de Treinamento Proposto

O uso de Redes Neurais em Sistemas de Controle, como já mencionado anteriormente, tem se mostrado como

uma ferramenta poderosa para manipular não-linearidades e outros problemas. Porém, para aplicar o algoritmo em sistemas reais, o tempo de treinamento se mostra como uma restrição importante.

O algoritmo de retro-propagação, embora muito simples de ser implementado, facilitando a simulação do sistema de controle proposto, consome muito tempo invalidando sua aplicação em tempo real. Não possui escalamento adequado, isto é, o tempo de treinamento aumenta exponencialmente na medida que o número de neurônios aumenta. Requer uma escolha adequada dos parâmetros de treinamento: taxa de treinamento, constante de momento e inclinação da função de ativação. E como um método de passos descendentes, pode parar em mínimo local. Estes são alguns dos inúmeros problemas associados com este algoritmo.

A despeito dos esforços para melhorar o desempenho da Regra Delta Generalizada com retro-propagação do erro, não existe ainda um método eficiente e confiável para treinar Perceptrons Multicamadas. O principal problema é o alto grau de não linearidade da função de erro.

Para contornar os problemas mencionados acima, um novo método para carregar a informação em um Perceptron Multicamada será apresentado. A idéia é usar aproximações quadráticas sucessivas para a função de erro, até chegar nas proximidades de um mínimo global. O algoritmo dos mínimos quadrados poderá ser utilizado na computação do mapeamento associativo ótimo. Este esquema associativo permite a inclusão seqüencial de novos padrões sem recalcular a expressão pseudoinversa para o conjunto completo de treinamento.

Na próxima sessão, o algoritmo proposto para Perceptrons Multicamadas será apresentado, o qual contorna dois grandes problemas: longo tempo de treinamento e algoritmo complexo de treinamento.

6- Técnica de Aprendizado Supervisionado

O método apresentado está baseado na Estimativa Ótima de Treinamento (OET)[5]. OET é uma técnica de aprendizado supervisionado para perceptrons multicamadas. Resultados preliminares indicam que a técnica OET pode ser muito mais rápida e precisa que outras técnicas.

O conjunto de padrões de entrada e saída são representados na forma de matrizes. Um mapeamento não-linear que associa padrões de treinamento de entrada com padrões de treinamento de saída, é encontrado resolvendo sucessivamente um sistema linear usando métodos diretos, que é de fato um procedimento baseado no método mínimos quadrados não-linear (aproximação local da função erro por uma função quadrática, e a exata aproximação da mesma por uma função aproximada). A pseudoinversa de Moore-Penrose é solucionada explicitamente em diferentes passos da OET. Uma importante vantagem desta técnica é que o problema do escalamento é muito bem definido. Baseado no número de operações envolvidas ($[rm]^2$ onde **r** é o número de padrões de treinamento de entrada e **m** a dimensão deles), é fácil estimar a taxa de aprendizado da OET com relação ao tamanho da rede.

A. Estimativa Ótima de Treinamento

A idéia geral da OET pode ser resumida usando a fig.6 [3]. Nesta figura hin,j (j=0,1, ..., m) é o conjunto das variáveis de entrada (hin,0 representa uma entrada adicional com valor constante igual a 1).

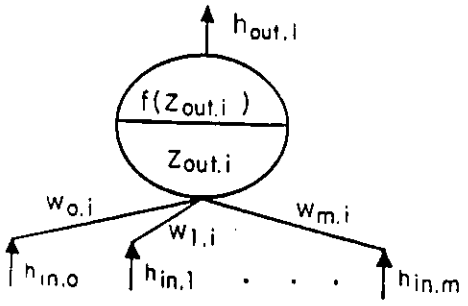


Fig 6 - Representação de um Neurônio

É bem conhecido que a propagação das entradas é feita pela soma das entradas ponderadas, isto é,

$$z_{out,i} = \sum_{j=0}^{m} h_{in,j} w_{ji}$$

O valor de hout,i é calculado pela aplicação da função de ativação hout,i = f(zout,i).

Para executar OET, um valor desejado (especificado) de saída hd-out,i, é retropropagado através da função de ativação e o resultado correspondente imediato desejado, zd-out,i , será obtido. Para executar esta operação, é necessario uma função contínua e inversível tal como a tangente hiperbólica. Esta função é adequada para OET, desde que as saídas desejadas possam ser positivas ou negativas. Assim, o conjunto das entradas e resultados intermendiários podem ser interrrelacionados para calcular a estimativa ótima, para um conjunto de interconecções ponderadas, no sentido dos mínimos quadrados.

A lista de símbolos utilizados para explicar o algoritmo será agora descrita (de acordo com a fig.2):

Hd-out -- matriz de saída desejada (r x n) ;
Hin -- matriz de entrada (r x m+1) ;
Zd-out -- **Hd-out** retro-propagado através da camada de saída (matriz r x n);
Wd-out -- matriz (m+1 x n) de interconecções ponderadas ligando a camada interna com a camada de saída;
Hd-hid -- matriz (r x m+1) ;
Zd-hid -- **Hd-hid** retro-propagado através da camada interna (matriz r x m+1);
Whid -- matriz (m+1 x m+1) de interconecções ponderadas ligando a camada de entrada com a camada interna ;
Zhid -- produzida pela multiplicação de **Hin** por **Whid** (matriz r x m+1);
Hhid -- **Zhid** propagado através da camada interna (matriz r x m+1);
Zout -- produzido pela multiplicação de **Hout** por **Wout** (matriz r x m+1);
Hout -- **Zout** propagado através da camada de saída (matriz r x n);
r -- número de padrões entrada/saída do conjunto de treinamento;
m+1 -- número de reurônios de entrada (que é iqual ao número de neurônios da camada interna); e
n -- número de neurônios de saída.

B. Passos de treinamento da Rede

O procedimento para treinamento de uma rede com uma camada interna será agora descrito. A extenção deste procedimento de treinamento para mais que uma camada interna é imediato.

1) Calcular o valor inicial de **Wout** (este deverá ser o valor final de **Wout** se camadas internas não são usadas):

$$W_{out} = H_{in}^{\Gamma} Z_{d-out}$$

onde

$$Z_{d-out} = \tanh^{-1}(H_{d-out})$$

e

$$H_{in}^{\Gamma} = (H_{in}^{t} H_{in})^{-1} H_{in}^{t}$$

assumindo r maior que m+1 (sistema sobre dimensionado). O símbolo Γ representa a inversa generalizada ou Moore-Penrose pseudo inversa.

2) Obter **Hd-hid** que irá produzir **Zd-out** dado **Wout** ,

- se n = m+1, $H_{d-hid}^{t} = (W_{out}^{t})^{-1} Z_{d-out}^{t}$

assumindo que W_{out}^{t} é não-singular

- se n > m+1 $H_{d-hid}^{t} = (W_{out}^{t})^{\Gamma} Z_{d-out}^{t}$.

$$(W_{out}^{t})^{\Gamma} = (W_{out} W_{out}^{t})^{-1} W_{out}$$

e

- se n < m+1 $H_{d-hid}^{t} = (W_{out}^{t})^{\Gamma} Z_{d-out}^{t}$

$$(W_{out}^{t})^{\Gamma} = W_{out} (W_{out}^{t} W_{out})^{-1}$$

caso indeterminado.

3) Normalizar os elementos de **Hd-hid** para garantir que eles irão se situar dentro do limite estipulado para as saídas das funções de ativação da camada interna,

Hd-hid = **Hd-hid** 0.99/ $|\lambda|$

onde λ é o elemento de **Hd-hid** com maior amplitude. Um fator de multiplicação, em torno de 0.99, é usado para obter valores finitos de **Zd-hid** , e tirar vantagem da não-linearidade da tangente hiperbólica.

4) Produzir **Zd-hid** usando a inversa da tangente hiperbólica,

$$Z_{d-hid} = \tanh^{-1}(H_{d-hid})$$

5) Dado **Hin** e **Zd-hid** , calcular **Whid** ,

$$W_{hid} = H_{in}^{\Gamma} Z_{d-hid}$$

6) Obter o valor da saída da camada interna;

$$Z_{hid} = H_{in} W_{hid}$$

e

$$H_{hid} = \tanh(Z_{hid})$$

7) Recalcular **Wout** ,

$$W_{out} = H_{hid}^{\Gamma} Z_{d-out}$$

onde

$$H_{hid}^{\Gamma} = (H_{hid}^{t} H_{hid})^{-1} H_{hid}^{t}$$

O treinamento da rede será finalizado no passo 7), isto é, uma vez as matrizes de interconecções ponderadas **Whid** e **Wout** estejam ótimamente calculadas usando o critério dos mínimos quadrados.

Para obter a saída do ANN para um dado conjunto de entradas (classificação/operação on-line), os seguintes passos são necessários:

1) Combinar **Hin** e **Whid** para produzir **Zhid** ,

Zhid = **Hin** **Whid**

2) Tratar **Zhid** com a função de ativação (tangente hiperbólica),

$$H_{hid} = \tanh(Z_{hid})$$

3) Combinar **Hhid** e **Wout** para produzir **Zout**

Zout = **Hhid** **Wout**

4) Tratar **Zout** com a função de ativação para gerar **Hout** (saída da rede),

$$H_{out} = K \tanh(Z_{out})$$

onde K=1.0101 (fator de normalização da saída (1/0.99)).

7- Conclusão

Rede Neural Artificial (ANN) possui importantes vantagens, as quais são de bastante utilidade na modelagem e controle de sistemas não-lineares[2]. A capacidade de aprendizado, generalização, paralelismo intrínceco, e capacidade de aproximar qualquer sistema não-linear, são algumas destas vantagens.

Devido aos avanços em ANN, é possível no momento o uso deste tipo de sistema inteligente em aplicações em tempo real. Uma das áreas da tecnologia que mais se beneficiaram disto é a de sistema de controle. O longo tempo, geralmente requerido, para o treinamento da ANN foi por muito tempo um problema crítico para utilização desta tecnologia em tempo real.

A Regra Delta Generalizada com retro-propagação do erro é o algoritmo mais usado em aplicações de controle. Embora muito trabalho tenha sido feito para melhorar o desempenho deste algoritmo de aprendizado, não existe ainda um método eficiente para treinamento de perceptrons multicamadas. O principal problema é o alto grau de não-linearidade da função de erro.

Inicialmente, a rede foi utilizada para reproduzir o comportamento entrada/saída do processo, isto é, um conversor tiristorizado controlando um motor DC com excitação independente. Então, uma segunda rede foi treinada para ser o controlador que irá manipular a energia fornecida ao motor para se obter a resposta desejada (representada pelo sinal de referência). Como o modelo foi identificado por uma rede neural, o erro final é retro-propagado através do modelo afim de se obter um erro equivalente para treinamento do controlador neural. Ambos, o controlador neural e a rede identificadora aprende continuamente durante a operação normal, pela monitoração da informação vinda dos sensores e relacionando com a saida desejada.

8- Agradecimentos

Este trabalho foi possível devido a ajuda do CNPq - Conselho Nacional de Desenvolvimento Científico e Tecnológico e FAPEMIG - Fundação de Amparo a Pesquisa de Minas Gerais.

9- Referências

[1] D.E. Rumelhart, G.E. Hinton, and R.J. Williams - "Learning internal representations by error propagation in parallel distributed processing", Exploration in the Microestructure of Cognition, Vol.1, Foundations, MIT Press, 1986, pp.318-362.

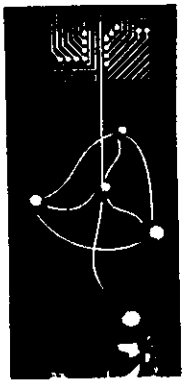
[2] K.J. Hunt, D. Sbarbaro, R. Zbikowski, and P.J. Galwtrop - "Neural Networks for Control System - A Survey" Automatica, vol.28, no.6, pp. 1083-1113, 1992.

[3] A.P. Alves da Silva, A.M. Leite Silva, J.C.S. de Souza, and M.B. do Coutto Filho - "State Forecasting Based on Artificial Neural", 11th Power Systems Computation Conference, PSCC, Avignon, August 1993.

[4] L.E. Borges da Silva, G. Lambert-Torres, E.C. Saturno, A.P. Alves da Silva, and G. Olivier - "Neural Net Adaptive Schemes for DC Motor Drives", IEEE Industry Applications Society Conference, Toronto, October 1993.

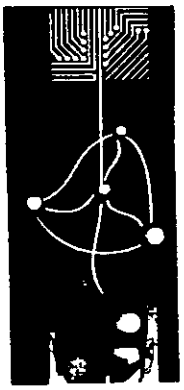
[5] J.F. Shepanski- "Fast learning in Artificial Neural Systems: Multilayer Perseptron Training using Optimal Estimation ", Proc. IEEE 2nd Intern. Conf. Neural Nets, San Diego, Jul. 1988, Vol.1, pp.465-472.

Anotações



CONTROLE II

Anotações



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajuba. 24 a 27 de outubro de 1994

Autonomous Control Using Soft Computing Paradigms

M. De Oliveira **M. Figueiredo** **F. Gomide**
UNICAMP/FEE/DCA
Campinas, São Paulo, Brasil

L. Romero
Universidad de Valladolid/Faculdad de Ciencias
47011 - Valladolid, Spain

assfalk@fee.unicamp.br mauri@dca.fee.unicamp.br
gomide@dca.fee.unicamp.br
luis@venus.isa.cie.uva.es

Abstract

In this paper, a genetic algorithm has been used to design of neurofuzzy systems as an alternative to traditional nonsupervised learning methods for neural networks. Fuzzy systems and neural networks also get improved behaviour with their symbioses. Here, we have integrated neural networks, fuzzy systems and evolutionary computing techniques to developed a control system for an automatically guided vehicle (AGV). The network learns to guide the AGV to given goals without collisions. The knowledge learned by the network can be extracted as if-then fuzzy rules and the number of rules, number of partitions and the shape and position of membership functions are easily determined.

1. Introduction

Recently, researchers have studied the symbiosis of genetic algorithm techniques, fuzzy set theory and neural networks. For instance, genetic algorithms have been used to design fuzzy systems.This determines membership functions, consequent parts and the number of *if-then* fuzzy rules [Tak93a]. Likewise fuzzy set theory is being used to adjust some parameters of genetic algorithm such as population size, mutation rate and crossover rate [Tak93b].

Another interesting integration concerns fuzzy set theory and neural networks [Lin91]. The resulting system is a neurofuzzy network, which shares the benefits of the original systems: parallel computation, robustness, natural language processing and imprecise data processing [Fig93].

Genetic algorithms have been considered as an alternative learning method for neural networks, filling the nonsupervised learning method gap [Ich92, Mac92].

In this paper we present a controller for an automatically guided vehicle (AGV)

which integrates the above three areas, that is, by using techniques of soft computing. The network learns to guide the AGV to goals without colliding with environment obstacles. The knowledge acquired by the network can be extracted as *if-then* fuzzy rules since it also determines the number of rules, membership functions and consequent parts. This neurofuzzy model is an attractive and convenient approach for designing fuzzy systems.

2. A Neurofuzzy Network

The fuzzy neural network used here will be briefly described below. For more details please refer to [Fig93], where it first appeared. The structure of the system under discussion will be centered around a set of "if-then" conditional statements (rules) given as follows:

input premise: X_1 is A_1 and X_M is A_M

rule 1: if X_1 is A_1^1 and X_M is A_M^1 then y is g^1

rule M: if X_1 is A_1^N and X_M is A_M^N then y is g^N

consequence: y is g

where: X_j is a fuzzy variable, A_j and A_j^i are the corresponding fuzzy sets, while "y" is a real variable, g^i are viewed as constants defined in the output space.

All the universes are assumed to be discrete. To simplify the notation we make z_k the grade of membership of Z at $x_k(Z(x_k) = z_k)$ with x_k denoting a numerical value in the input space.

The control decision y is determined via a sequence of the three reasoning stages: matching, antecedent aggregation and rule aggregation.

These steps are carried out by specialized neurons, the min-max neurons [Gom92], which model more closely their biological counterparts, by incorporating more complex decodification function, synaptical and input aggregation operators [Roc92].

The proposed fuzzy neural network (see Fig. 2.1) has a feedforward architecture with five layers of neurons. The first layer implements input fuzzification, the second rule antecedent matching, the third antecedent aggregation, the fourth consequent aggregation. The last layer supplies the final output, as a weighted average of each rule output contribution.

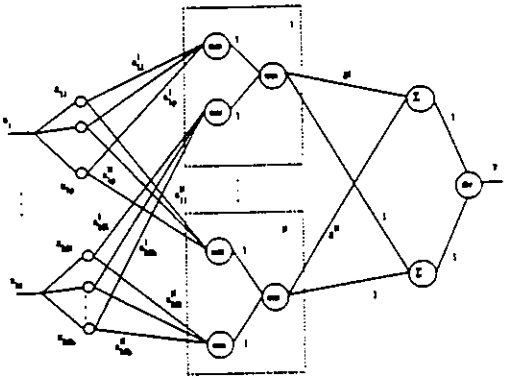


Figure 2.1: The fuzzy neural network.

3.A Brief Overview of GA

A genetical algorithm (GA) is basically a search procedure based on biological evolutionary mechanisms: natural selection, genetic recombination and mutation.

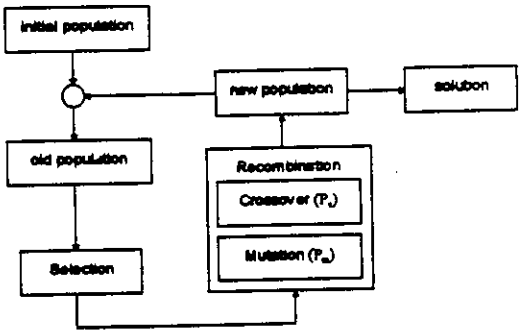


Figure 3.1 The basic Genetic Algorithm.

As in its natural counterpart, the GA operates upon a population of candidate solutions, a sample of points in the solution-space. Each individual is represented by its operational characteristics, in a coded form, the *chromosome*. This chromosome may be thought of as the coordinates of the

individuals in the solution space. The chromosome may be a string of bits (most common), of real or integer numbers or even an array of numbers (as in [Ich92]).

The search procedure is outlined in Figure 3.1. At first, the whole population is evaluated singly and each individual is awarded a *fitness* function, based upon its performance as a solution. Then, a set of individuals (usually two) are selected, according to their fitness function, for the re-combination operation. Recombination consists of two operations: *crossover* and *mutation*. These operations occur with probabilities P_c and P_m, respectively.

Crossover promotes the exchange of parts of the involved individuals' chromosome. One or more crossover sites are selected and the genetic substrings thus defined are switched among the individuals (see Fig. 3.2). The principle (the *building blocks* principle) behind this exchange of genetic material is that high fitness individuals owe their good performance to good characteristics coded in their chromosomes. Thus the recombination of good building blocks would result in better individuals. The mutation operator adds a variable amount of noise to the chromosome, generating new varieties of solutions.

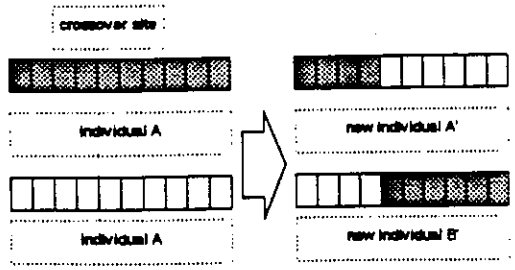


Figure 3.2 Single Site Crossover.

The selection and recombination process is repeated until a given amount of new candidate solutions are generated. This percentage, the *generation gap (G)*, is usually defined in terms of a fraction of the size of the original population. A new population is formed by (1-G)*sizeof(old population) individuals of the old population plus the newly-generated individuals. A variation of this scheme involves keeping the best individual of the old population. This is known as *elitism*, and usually is used with high gap ($G \rightarrow 1$) variations of the GA.

Once a new population is formed, the process is repeated, until a satisfactory solution is created or until a given number of population iterations, known as *generations*, is executed.

It can be shown that the population maintained by the GA converges to an optimal solution [Qi94].

4.Neurofuzzy learning employing a GA

The design of neural networks using GA techniques has been considered by many authors. At present, three main research fields may be identified: the determination of the overall structure of the neural network (number of layers and neurons, as well as the connections between them), the optimization of traditional training algorithm parameters and the optimization of the synaptic weights.

Ichikawa and Sawa's use of GA [Ich92] for training a neural-network is an example of the synaptic-type GA-neural network integration. They use an integer array-codified neural network as a population member. The characteristics encoded were the synaptic weights, while the structure remained the same throughout each case problem. Since their neural network was inserted in a feedback control loop, they could not rely upon the usual (input, output) comparison fitness function. Instead they built a

fitness function that evaluated the behaviour of the controlled plant during the simulation, against that what would be desired.

Harp et al. [Har89]took the overall structure track and developed a GA software package that tuned both the backpropagation parameters and the structural characteristics of the neural netowrk. Once again, the feed-forward network model was used.

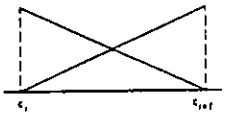
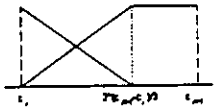
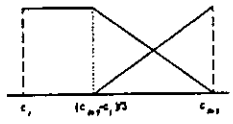
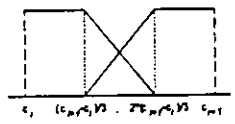
In our approach both the synaptic-optimization and structure determination are applied to the neurofuzzy network.

The GA employed is an elitist model operating upon an integer/real chromosome. The characteristics tuned by the GA over the generations are the rule base consequents and the membership functions, given by shape and position. The rules consequents are represented as real genes, while the membership function representation is more complex, contained in a 2-tuple (center, type), where center determines the whereabout of the function centre point and type is either trapezoid shape or triangular shape (see Table 3.1). Thus, for an n-rule rule base, k antecedents and f_i membership functions per antecedent, the chromosome describing this neurofuzzy network has $n+\sum_{i=1}^{k} f_i$ real genes and $\sum_{i=1}^{k} f_i$ integer genes.

The initial population is generated randomly. The membership function genes are interpreted according to its shape and center point and its immediate neighbours' shapes and center points. The previous function determines the left side of the current function and the following function the right side of the current function. See Table 3.1.

Crossover always occurs, selecting only one site. The individual selection uses the weighted roulette algorithm, which confers higher selection probabilities to individuals with higher fitness. The mutation operator is quiescent for most of the run (probability zero).

Table 3.1 Membership Function Shaping

Function (i)'s shape (center point is c_i)	Function $(i+1)$'s shape (center point is c_{i+1})	Function (i)'s right side and Function $(i+1)$'s left side
Triangular	Triangular	
Triangular	Trapezoid	
Trapezoid	Triangular	
Trapezoid	Trapezoid	

The mutation probability jumps abruptly to 0.1, for one generation, when the average population fitness becomes greater than or equal to 90% of the best population fitness. This population "shake-up" creates individuals with new genetic traits, staving off population staleness. Our convergence-triggered mutation operator is based on Kauffman's strategy of periodic mutational bursting [Kau86].

Each individual was evaluated by placing it in a simulated environment containing obstacles and targets for a given amount of time or until it collided

with an obstacle. The objective function is given as a function of the number of targets reached, the mean distance covered and the distance to the target at the end of a simulation. The fitness function value is generated by scaling the population objective function values.

5. Results

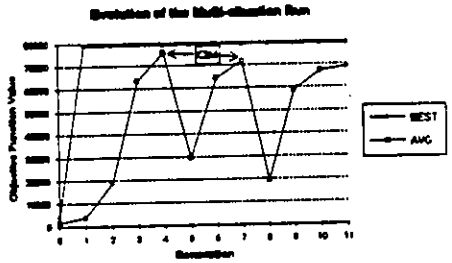


Figure 5.1 Objective Function Evolution,Run A.

Two variants of the GA described in section 4 were used. Variant A had a 104 member population and used a simulation with 3500 time steps and presenting 4 obstacles in 4 sequences ((A,B,C,D), (B,C,D,A) etc.). The objective function was taken as an average of the objective functions over each sequence.

Variant B had a 64 member population and used a simulation with 2000 time steps, presenting only one sequence with four obstacles (A,B,C,D). The convergence behaviour can be seen in Fig.5.1.

Variant A presented the best results, given its more thorough and varied objective function evaluation (Fig.5.2). The AGV was able to navigate successfully even when the targets were placed in different places (Fig.5.3).

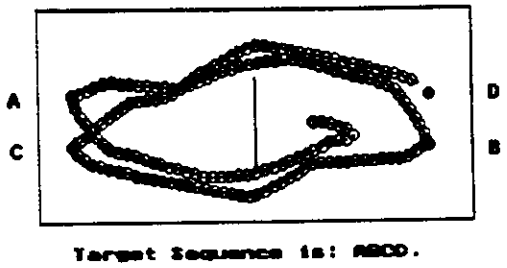


Figure 5.2 Best individual run, Variant A.

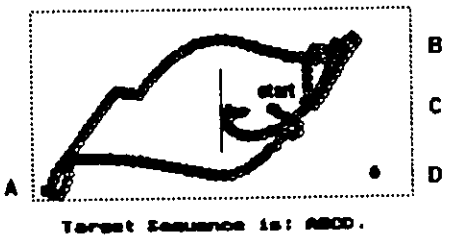


Figure 5.3 Different targets, Variant A

6. Conclusion

We introduced in this paper a type of genetic algorithm as an alternative learning method for neurofuzzy networks. The method was evaluated using the AGV autonomous control problem. Simulation results show that the network successfully guides the vehicle to the goals without colliding.

We have also investigated the control of mutation rate using the degree of population convergence. Mutation occurs only if the population convergence is greater than a given threshold value.

After the learning period we were able to extract the knowledge of the network as if-then fuzzy rules (number of rules, partitions and membership functions).

Currently we are seeking to progressively increase the complexity of the problem, using two approaches:

- incorporating a set of several, increasingly complex environments in the evaluation;

• implementing several runs, each with increasingly difficult lay-outs, where a part of the initial population of each run would be taken from a sample of the preceding run.

Acknowledgements

The authors would like to acknowledge CAPES, FAPESP 93/3034-8, CNPq #300729/86-3 and ESPRIT-ECLA005 for their support.

References

[Fig93] M. Figueiredo, F. Gomide and W. Pedrycz, "A Fuzzy Neural Network: Structure and Learning", 5th IFSA World Congress, Seul, Korea, 1993.

[Gom92] F. Gomide and A. Rocha, "A Neurofuzzy Components Based on Threshold", IFAC, Silica, Spain, 1992.

[Gol89] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, 1989.

[Har89] S. Harp, T. Samad and A. Guha, "Toward Genetical Synthesis of Neural Networks", *Proceedings of ICGA'89*, pp.360-369.

[Ich92] Y. Ichikawa and T. Sawa "Neural Network Application for Direct Feedback Controllers", IEEE Transactions on Neural Networks, Vol.3. No.2, March 1992.

[Kau86] S.A. Kauffman and R. Smith "Adaptive Automata Based on Darwinian Selection", Physica 22D, pp. 68-82, 1986.

[Lin91] C.-T. Lin and C.S.G. Lee,"Neural-Network-Based Fuzzy Logic Control and Decision System", IEEE Transactions on Computers, vol.40, no.12, December, 1991.

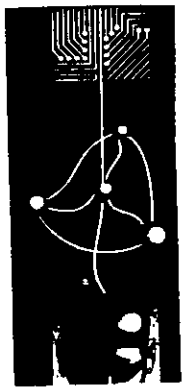
[Mac92] R. Machado and A. Rocha, "Evolutive Fuzzy Networks", Proc. of the International Conference on Fuzzy Systems, pp.493-500, San Diego, USA, 1992.

[Qi94] X. Qi and F. Palmieri, "Theoretical Analysis of Evolutionary Algorithms With a Infinite Population Size in Continuous Space, Parts I & II", IEEE Transactions on Neural Networks, Vol.5, No.1, pp.102-129, Jan.1994.

[Roc92] A.F. Rocha, Neural Nets: a theory for brains and machines, Springer Verlag, 1992.

[Tak93a] H.Takagi, "Neural Networks and Genetic Algorithm Approaches to auto-design of Fuzzy Sytems", Proc. of Fuzzy Logic in Artificial Intelligence, Linz, Austria, June, 1993.

[Tak93b] H. Takagi, "Dynamic Control of Genetic Algorithms using Fuzzy Logic Techniques", Proceedings of the 5th Int'l Conf. on Genetic Algorithms, Urbana-Champaign, IL, July,1993.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá. 24 a 27 de outubro de 1994

The Use of Genetic Algorithms for the Evaluation of Inverse Kinematics of Manipulators

Carlos Henrique da Silveira
(silveira@dcc.ufmg.br)

Lúcio de Souza Coelho
(omni@dcc.ufmg.br)

Mário Fernando Montenegro Campos
(mario@dcc.ufmg.br)

May, 1994

Departamento de Ciência da Computação
Instituto de Ciências Exatas
Universidade Federal de Minas Gerais
Caixa Postal 702 – Belo Horizonte – MG – Brazil – CEP 30.161-970

Abstract

This paper describes the implementation of a genetic algorithm system able to compute the inverse kinematics of a generic manipulator (robot). Given its geometric description and an homogeneous transformation describing the goal to be reached by its end-effector, the system computes the joint angle/displacement values with the desired precision. The method presented good performance in the case of a 6 degree of freedom manipulator, with a fast convergence to the specified values. An application to the case of redundant manipulators is also presented, where analytical solution by traditional methods would be of high complexity.

1 Introduction

The inverse kinematics problem can be stated as, given a position and an orientation in the workspace of a manipulator, find the set of joint values that will accomplish them. This problem is complex due to the multiplicity of solutions and to the difficulty to obtain, in the generic case, analytical solutions to the non-linear equation systems for six or more degrees of freedom manipulators [1].

The approach to non-linear equation system is, in general, a high-complexity problem. For this reason alternative methodologies have been widely researched in the field of Artificial Intelligence. Particularly Genetic Algorithms (GA) [2-7] have show to be an useful tool for some complex applications such as optimization and machine learning.

It is described here a system which has developed to generate the joint values, given the description of a manipulator in terms of its geometrical parameters and a final position and orientation of the end-effector. The values to be assumed for the joint variable parameters are computed so that the manipulator reaches the goal (or a transformation) specified by the user, with

a given precision

2 The Genetic Algorithm

2.1 Mapping Between Individuals and Solutions

In the GA solution for the inverse kinematics problem, individuals are sets of joint values. As strings of 1's an 0's, each of those sets is represented by the concatenation of substrings of equal length, each of them corresponding to a joint value; each substring is interpreted as a Gray Code number [1]; the value V of a given joint is obtained from

$$V = n\frac{M}{N} + I$$

where n is the integer number represented by the substring, M is the length of the range where the joint value can vary, N is the greatest number representable by a substring and I is the lower joint limit. For example, imagine a very simple manipulator composed of only three revolute joints, each of them able to rotate from 0° to 180°. Suppose that four bits are used to represent each joint and we want to find out to which configuration corresponds the individual

0010 1110 0101.

The first substring (from left to right) of four bits (0010) corresponds to the first joint, the second substring to the second and so on. Making the conversion for the second joint using the formula above and taking as parameters $n = 3$ (0010 in Gray Code), $M = 180°$, $N = 15$ (the largest number represented with four bits) and $I = 0°$, we find that $V = 36°$. Computing V for the other joints and using the geometrical notation suggested by Denavit-Hartenberg [1], we find that individual corresponds to

$$\theta_1 = 36°, \theta_2 = 132° \text{ e } \theta_3 = 72°.$$

[1] Gray code was selected to avoid disruptive mutations, as suggested in [2].

2.2 Fitness Evaluation

A fitness function is chosen such that the precision criterion selected is met. It is the following: let be A the homogeneous transformation describing the end-effector of a manipulator M under its frame 0 when the joints assume the values with respect to an individual x: let B be the homogeneous transformation that is desired to be assumed by the end-effector under the frame 0; making $C = A - B$ and calling i and j the coordinates of the C element with the greatest absolute value among all, its is said that x satisfy a precision p if $|c_{i,j}| \leq p$. Keeping in mind this definition, we use the fitness function:

$$f(x) = \frac{1}{|c_{i,j}|}$$

2.3 About the Reproduction and the Population

Individuals with the best fitness have a greater probability to reproduce in GA. In order to satisfy this paradigm, we choose the following distribution. Imagine a population with N individuals in decreasing order of fitness, that is, $f(n-1) \geq f(n) \geq f(n+1)$. To select an individual to reproduce among the population members, we use the function

$$s(r) = r \bmod (r \bmod N + 1) + 1$$

where $r \in \mathbb{N}$ is a random number. Let S_n be the event of the individual n to be selected to reproduce (that is, $s(r)$ returns n). It follows that

$$P(S_n) = \frac{1}{N}\sum_{i=n}^{N}\frac{1}{i}$$

The probability distribution is shown in Figure 1 for $N = 65$.

It should be pointed out that two individuals are always selected to mate. The first individual is straightforwardly chosen by $s(r)$. But, when selecting the second,

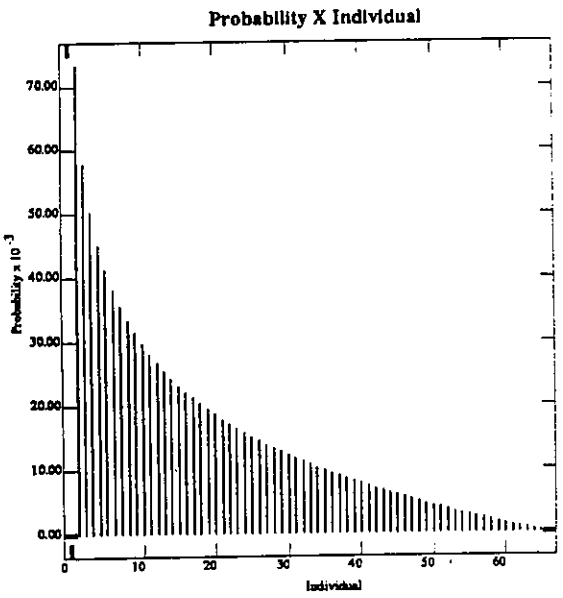


Figure 1: Probability distribution (vertical axis) *versus* individuals in decreasing order of fitness (horizontal axis).

it must avoided selecting the first individual. So, the first individual is skipped during the second selection, which implies that $s(r)$ works with a population smaller by one unit.

When mating two individuals, the two offspring generated by a crossover are placed in the next generation. Each offspring still experiences a mutation, which is implemented by the complement of a randomly selected bit.

The new generation grows until its size becomes equal to the size of the current generation, hence, the population held constant.

Finally, we choose to follow the elitist model as in [5]: the best fitness individual is cloned; a copy of it is always placed in the next generation. Thus, it is ensured that, in the running the GA, a "retreat" never occurs in the search for a satisfactory solution.

2.4 The Crossover Operator

When the character subsets, that are interchanged among two individuals by crossover, are equal, the offspring generated are only copies of their parents. To avoid this, the crossover operator used here first determines what can be called an *useful crossover region*: which delimited by the leftmost and rightmost different bits comparing two strings selected for reproduction. For example, consider the pair of individuals below, where the useful crossover region is shown underlined:

100110010010
and
100110101010.

Then, a crossover is made by selecting a point between the bits belonging to the useful crossover region, ensuring that the offspring will be different from its parents, as in

100110101**1010**
e
100110010010.

The useful crossover concept helped us to think what to do when the two individuals that are about to reproduce are equal or different only by one bit, in which case there is no useful crossover point. To avoid this situation one of the individuals will be forced to undergo 1 or 2 mutations, depending if they differ by 1 or 0 bits, respectively. Then, a useful crossover is made.

3 Results

3.1 Case Study I

In this section the speed of the current GA is analyzed statistically for the problem of the inverse kinematics of a manipulator.

A single manipulator is used, composed by three prismatic joints (perpendicular among them and responsible for the positioning of the manipulator) and three final revolute joints (responsible for the orientation of the end effector, with coincident origins and perpendicular rotation axes).The Denavit-Hartenberg parameters

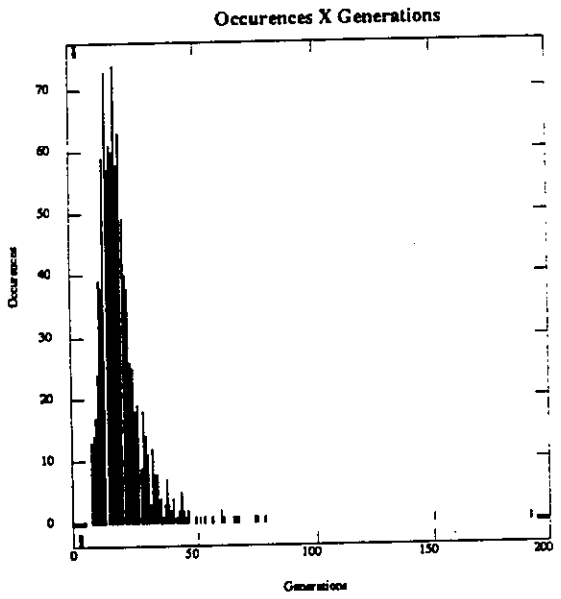


Figure 2: Number of occurrences (vertical axis) *versus* generations number (horizontal axis).

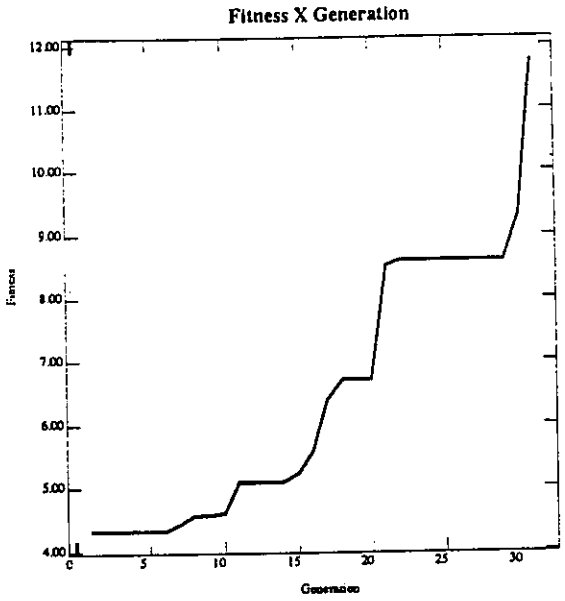


Figure 3: Best individual fitness (vertical axis) *versus* generation (horizontal axis) for the discovery of a solution to a six degree of freedom manipulator.

for the mentioned manipulator are shown in the table below:

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	$0°$	0	d_1	0
2	$90°$	0	d_2	$90°$
3	$-90°$	0	d_3	0
4	$0°$	0	0	θ_4
5	$-90°$	0	0	θ_5
6	$90°$	0	0	θ_6

For simplicity, it was assumed that prismatic joints can vary their values continuously from 0 to 1 and revolute joints have no rotation limits. Finally, the transformation that describes the end-effector with reference to the frame of the last link is the identity matrix.

Wishing only to evaluate the speed of the algorithm, we work with a little precision of 0.1, and 6 bits were designated to represent each joint value. Therefore, our individuals were strings with length 36 over the alphabet $\{0, 1\}$. Population size was arbitrarily assigned to 65 individuals (1 individual cloned and 32 pairs generated by reproduction).

Each test was made executing the following steps:

1. Random values, within the variation bounds specified in manipulator description, are generated to the joints.

2. Using the direct kinematics [1], a transformation A, describing the end-effector with reference to the frame 0 when joints assume the values generated in Step 1, is obtained. Therefore, A surely is in the manipulator workspace.

3. It is asked to the algorithm to supply the joint values that reach A with the specified precision.

4. The result of the test is the number g of generations spent by the algorithm to reach the solution.

Data from executing 1000 tests is obtained. Figure 2 shows the histogram with the frequency distribution for each test result. The average number of generations was 19, with 5 and 193 as the minimum and the maximum, respectively. In order to give a notion of the real time spent.

each test took, in average, about one hundredth of a second of CPU time running on a SPARC2 workstation.

Figure 3 shows the fitness evolution through generations in one test.

3.2 Case Study II: a Manipulator with Redundant Degrees of Freedom

The greater the number of joints of a manipulator, the more difficult and laborious it is to obtain its inverse kinematics equations. Manipulators normally have only six joints, each one corresponding to a degree of freedom, which is the minimum needed to ensure a non-zero dextrous workspace. When a manipulator has more than six degrees of freedom, it is called redundant, because there are redundant degrees of freedom.

Even though it is hard to obtain analytically, the inverse kinematics of a redundant manipulator can be obtained automatically by the method presented here. That gives manipulator independent solutions for this problem.

In order to test this generality, it was defined a manipulator much more complex than the one of the previous section, having 10 degrees of freedom and only revolute joints. We assume that the odd link joints do not have bounds restricting its rotations and the even link joints can rotate in the range of 0° to 180°. The transformation that describes the end effector in relation to last link frame is the identity matrix. The manipulator is described below according to Denavit-Hartenberg notation:

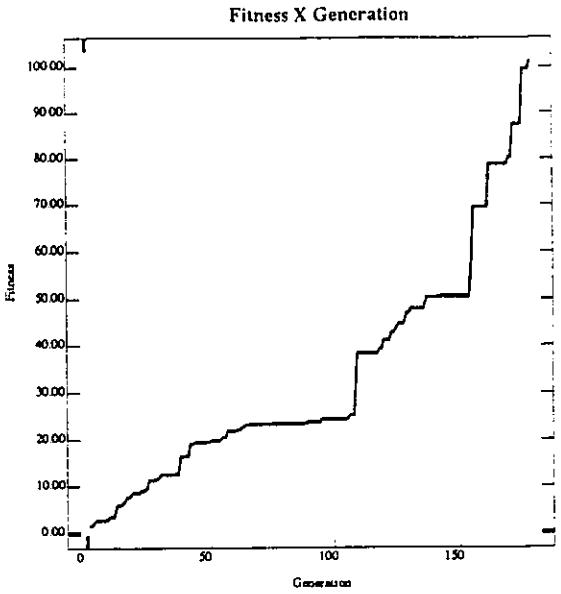


Figure 4: Best individual fitness (vertical axis) *versus* generation (horizontal axis) for the redundant manipulator.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0.4	θ_1
2	-90^o	0	0	θ_2
3	-90^o	0	0.4	θ_3
4	-90^o	0	0	θ_4
5	-90^o	0	0.4	θ_5
6	-90^o	0	0	θ_6
7	-90^o	0	0.4	θ_7
8	-90^o	0	0	θ_8
9	-90^o	0	0.4	θ_9
10	-90^o	0	0	θ_{10}

The GA worked with a precision of 0.01 and used 10 bits to represent each joint value. A population of 65 individuals was used. The manipulator reaches the identity matrix, which means that the end-effector frame reached the position and orientation of the base frame within the required precision. The matrix below was produced as the solution

$$\begin{bmatrix} 1.000 & -0.004 & 0.005 & 0.005 \\ 0.004 & 1.000 & 0.008 & 0.010 \\ -0.005 & -0.008 & 1.000 & 0.009 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{bmatrix}$$

in 179 generations. The fitness evolution of the best individual during generations is shown in Figure 4.

4 Conclusions

The method presented here worked for the tested cases, functioning as a generic evaluator of inverse kinematics. Although, we consider the tests very simple and, hoping to know more about the behavior of this GA, it would be convenient pursue the following directions: verification of bit convergence, variation of population size, variation of crossover rate, variation of mutation rate, a study of the relations between the bit number per joint and precision and the use of other probability distributions related to individual selection for reproduction.

5 References

1 Craig, John J., **Introduction to robotics: mechanics and control.** 2nd edition, Addison-Wesley Publishing Company, Inc.

2 Forrest, Stephanie, **Genetic Algorithms: Principles of Natural Selection Applied to Computation.** in *Science*, Vol. 261, 13 Aug. 1993. pp. 872-8.

3 Goldberg, David E., **The Genetic Algorithm Approach: Why, How, and What Next?**, in *Adaptative and Learning Systems - Theory and Applications*, Plenum Press, New York and London, pp. 247-53.

4 Holland, John H., **Genetic Algorithms**, in *Scientific American*, July 1992, pp. 44-50.

5 McGarrah, D.B. e Judson, R.S., **Analysis of the Genetic Algorithm Method of Molecular Conformation Determination**, in *Journal of Computational Chemistry*, Vol. 14, 11, 1993, pp. 1385-95.

6 Unger, Ron e Moult, John, **Genetic Algorithms for Protein Folding Simulations**, in *Journal of Molecular Biology*, 231, pp. 75-81.

7 Wayner, Peter, **Genetic Algorithms**, in *Byte*, January 1991, pp. 361-8.

1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajuba
Itajuba, 24 a 27 de outubro de 1994

Sistema Inteligente para Problemas de Gas-Lift

Antonio R. Patrício, DEPRO/PETROBRÁS
Armando F. Rocha, UNICAMP
Celso K. Morooka, UNICAMP

RESUMO

A produção de óleo e gas natural é muito sensível à perturbações no poço e/ou no processo. Identificação de problemas em poços de gas-lift e na planta de processo envolve também raciocínio simbólico, uma vez que as cartas de gas de superfície e os históricos de funcionamento do processo provêm vários dados considerados importantes, mas não todos requeridos para a tomada de decisão. Uma variedade de problemas de poço e planta pode ocorrer com o sistema de gas-lift. Redes neurais foram usadas para desenvolvimento do SEPLANT, um sistema especialista cujo objetivo é fornecer suporte ao controle inteligente das operações de gas-lift e de seu correspondente processo, servindo de ferramenta para os operadores nas tomadas de decisões em situações de contingência, de forma a evitar problemas e perdas de produção.

INTRODUÇÃO

Na explotação de campos de produção de óleo e gas natural, a vida útil dos poços e do processo pode ser considerada segundo as três seguintes fases(Homem, 1991):

. período quando a produção somente aumenta, até alcançar seus valores máximos. Durante esta fase, o reservatório tem pressão suficiente para trazer óleo até a planta de processo na superfície;

. período quando a pressão do reservatório começa a declinar, sendo então necessário o uso de elevação artificial para manter a produção do reservatório;

. período quando os sistemas de injeção de água e/ou gas natural começam a operar, dependendo dos resultados dos estudos e do comportamento da performance do reservatório.

A principal etapa da seleção de um sistema de elevação artificial é a previsão da vazão esperada para cada método analisado. Esta vazão pode ser determinada pelo traçado das curvas de IPR e OPR, que representam a habilidade de produção de fluidos dos poços. Estas curvas relacionam a pressão disponível do reservatório com a vazão e a pressão requeridas na entrada do tubing, para elevar o líquido até a superfície e são usadas também para estimar qual o aumento desta vazão para um dado método de elevação artificial.

A tecnologia de elevação por gas-lift vem evoluindo ao longo dos últimos anos, desde o início dos trabalhos pioneiros no começo da década de 1950. Este método de elevação situa-se em segundo lugar logo após o bombeio macânico, com base no número de instalações no mundo (Hasan e Kabir, 1993). Um sistema de gas-lift tipo simples, é composto de tubing, válvulas de gas-lift, mandris, packer, separador, compressor e instrumentação. O

gas é comprimido na superfície e injetado no tubing através das válvulas de gas-lift, para aerar a coluna de líquido que está sendo produzida, reduzindo assim a pressão de fundo em fluxo permitindo uma produção maior de líquido. Avanço rápido tem sido observado no projeto dos vários tipos de instalações otimizadas de processo, em particular as otimizações que tornam esta técnica mais atrativa, principalmente para poços marítimos.

Variáveis como razão gas-líquido, vazão de gas produzido e injetado, pressões de revestimento e de tubing, e as cartas de gas de superfície (registradores) são peças importantes para operação de gas-lift.

O sistema de gas-lift tem operação mais complexa que o sistema de surgência natural pois a razão gas-líquido é variável. O conhecimento preciso do volume de gas injetado não é uma informação confiável para o controle de processo, porque este volume depende da pressão na profundidade de injeção.

Problemas como malfuncionamento de válvulas (poços e processo), vazamento de mandril, e outras condições anormais de processo, pode ocorrer com este sistema (Brown, 1980).

A identificação de problemas em sistema de gas-lift envolve também raciocínio simbólico, desde que a instrumentação convencional e as cartas de gas de superfície fornecem várias informações importantes, mas não todas requeridas para a tomada de decisão (Patricio, 1992).

Técnicas de inteligência artificial estão emergindo como ferramentas adequadas para suporte à tomada de decisão na indústria do petróleo. Qualquer sistema inteligente especialmente projetado para supervisionar e controlar as atividades de gerenciamento da produção de poços e funcionamento do processo, deve ser capaz de executar o monitoramento inteligente usando o conhecimento de especialistas (Alegre et al., 1993). O objetivo deste trabalho é apresentar o SEPLANT, um sistema inteligente hierárquico, para ajuda ao engenheiro na operação de poços de gas-lift. Uma vez estabelecido o diagnóstico, as redes neurais podem também indicar a ação correta a ser tomada para manter as condições especificadas do gas-lift.

O NEURÔNIO ARTIFICIAL

Recentemente, grandes progressos foram denotados na neurociência e na ciência da computação renovando o interesse em redes neurais, como arquiteturas de uso potencial na resolução de problemas. Redes neurais são circuitos dinâmicos compostos de camadas de neurônios altamente interconectadas. Elas são únicas, devido sua habilidade em aprender com a experiência, como fazem as pessoas (Garcia e Whitman, 1992). Por exemplo, elas podem ser indicadas, com exibições repetidas, a reconhecer problemas de gas-lift e de seu processo correspondente. Muitos tipos de redes neurais têm sido propostas na literatura. Aqui o neurônio formal introduzido por Rocha,1992 é usado devido sua simplicidade e sua utilização efetiva em vários domínios tais como reconhecimento de padrão, e um certo número de aplicações petrolíferas.

(Rocha, 1992) propôs, no sentido de aumentar a capacidade de processamento do neurônio artificial, um novo modelo formal incluindo algumas propriedades emprestadas pelas redes Petri no neurônio original de Mc Culloch-Pitts. Este novo tipo de neurônio tem também a vantagem de descrever o conhecimento recente sobre a fisiologia da sinapse descoberta por biólogos. Foi mostrado (Fig. 1) que a transmissão dos neurônios pré e pós-sinapticos envolve a liberação especifica de mensagens químicas pré-sinapticas chamadas transmissores (t)para ligar moléculas pós-sinapticas definidas chamadas receptores (r). Esta ligação t'r ativa por sua vez, algumas outras moléculas pós-sinapticas chamadas controladores (c), cujo propósito é exercitar alguma ação de economia neuronal. Em outras palavras, a ativação de uma sinapse resulta em transações do tipo:

$$t + r \text{ ---}> c \text{ ---}> \text{ação} \qquad (1a)$$

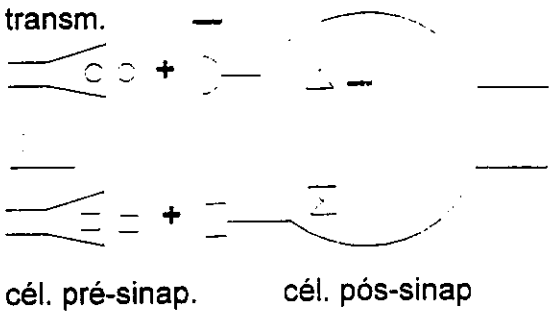


Fig. 1 O Neurônio Simbólico

A quantidade a(i,c) de controladores ativados pela i-ésima sinapse de um neurônio n_j, depende:

1b) da quantidade de a(i,r) de receptores r disponíveis no local pós-sinaptico;

1c) da quantidade a(i,t) de transmissores t liberados no terminal pré-sinaptico, e;

1d) da afinidade química $\mu(t,r)$ entre as moléculas pós-sinapticas t e r.

Agora:

2a) se A(i,t) é a quantidade total de transmissores t Armazenados no local pré-sinaptico e;

2b) se A(i,r) é a quantidade total de receptores armazenados pela célula pós sinaptica, então;

2c) o peso wi da sinapse é calculado como:

$$wi = A(i,t)\ \rho\ A(i,r)\ \lambda\ \mu(t,r)$$

onde ρ e λ são em geral, t-normas.

Nesta condição, o volume atual a(i,t) de transmissores liberados na i-ésima sinapse pode ser calculada como:

$$a(i,t) = s_j * w_i \qquad (3a)$$

onde * é, em geral, uma t-norma e o volume a(i,c) de controladores ativados pela ligação t/r pode ser assumida como:

$$a(i,c) = g(a(i,t)) \qquad (3b)$$

no caso, g é a função identidade:

$$a(i,c) = s_j * w_i \qquad (3c)$$

se m fontes s_i provêm a informação de entrada de um neurônio n_j com relevância r_i cada, então sua saida s_j é obtida como:

$$a = \sum_{i=1}^{n} s_i * w_i \qquad (4a)$$

$$s_j = f(a) \qquad (4b)$$

onde s_j é uma função da média ponderada de suas entradas s_j. A função f é em geral uma função booleana tal que:

$$1 \text{ if } a > \alpha \qquad (4c)$$

$$s_j = 0 \qquad \text{otherwise}$$

mas ela pode ser definida como qualquer função de filtro. A relevância w_i é o peso da conexão (sinapse) entre o neurônio pré-sinaptico (fonte de informação) i e o neurônio pós-sinaptico n_j (meio de processamento). Procedimentos de aprendizagem ("backpropagation" é o mais famoso entre eles) mudam os pesos sinapticos de acordo com o sucesso (recompensa) ou falha (punição) de n_j para fornecer a solução do problema em estudo. nesta condição, w_i é uma medida de incerteza estatística da contribuição de s_i para a solução do problema.

Os neurônios são organizados em camadas para formar redes especializadas em resolução particularmente de problemas complexos. Os neurônios (sensoriais) da camada de entrada são indicados para comparar o valor atual de variáveis específicas v de conhecimento protípico v' sobre o comportamento destas variáveis, assim s_i provê uma medida de incerteza do "matching" $v \equiv v'$. Os neurônios das camadas intermediárias mediam e filtram as informações de entradas, equações 4a. a 4c. Estas operações provaram ser equivalentes aos cálculos envolvidos na resolução de sentença do tipo (Yager, 1990; Rocha, 1992) :

se Q (relevância Xs é A é verdade) então y é B (5)

Onde Q é um quantificador linguístico como a maioria de, se X de N e assim por diante.

Nesta condição a rede neural especializada (Fig. 2) é composta de :

1.nodos de evidências: são os nodos situados no nível mais baixo da rede. Eles representam os

dados que suportam as decisões nesta aplicação.

2.nodos intermediários: representam diferentes agregações de evidências usadas pelo especialista em seu raciocínio.

3.nodos de hipóteses : são os nodos situados no nível mais alto da rede e representam as hipóteses de diagnósticos levadas em consideração.

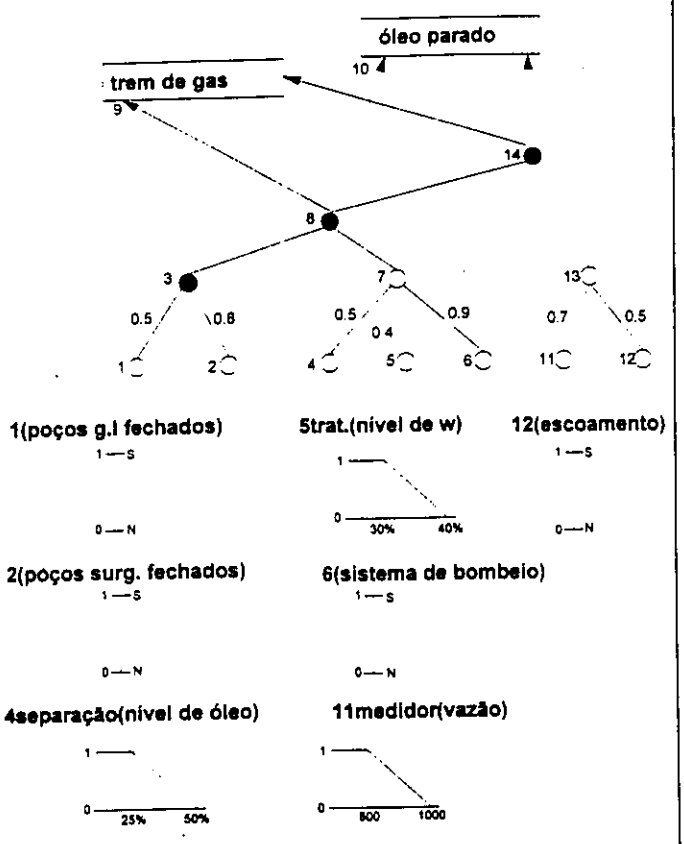


Fig. 2 Rede Neural

REDES NEURAIS

O conhecimento que o especialista adquire enquanto vivencia os problemas de seu domínio de especialização. é guardado em seu cérebro por uma troca de conectividade entre neurônios, e pelos neurônios especializados em executar operações definidas e cálculos lógicos selecionados (Rocha and Yager, 1992; Rocha, 1992). Devido a isto, o conhecimento do especialista pode ser representado como uma rede de conhecimento composta por um conjunto de redes interconectadas. cada rede codifica o raciocínio procedural usado pra suporte da tomada de decisão sobre os problemas do domínio.

Redes neurais foram construídas para imitar o raciocínio do especialista usado no controle do gas-lift e seu processo, sob condições operacionais definidas. Este conhecimento é representado como um conjunto de redes neurais relativo aos casos especificados do domínio que interessa ao engenheiro de campo. Todas as redes neurais são montadas para identificar problemas definidos de gas-lift e ajudar o engenheiro na tomada de decisão sobre a ação de controle requerida para manter a operação normal.

Cada rede neural é composta de um conjunto de nodos terminais chamados evidências (nodos 1,2,4,5,6,11 e 12 na Fig. 2), cuja tarefa é avaliar as condições requeridas para prover decisões operacionais. Estes nodos terminais contêm conhecimento prototípico sobre valores de variáveis controláveis que suportam tomadas de decisões operacionais especificadas (gráficos de funções da Fig. 2). Os valores atuais destas variáveis são "matched" contra o conhecimento prototípico e determinam o gráu de ativação do nodo terminal correspondente. Este gráu de ativação representa a confiança dos dados contra a evidência prototípica requerida para a tomada de decisão.

Os resultados terminais são "clustered" em nodos não terminais chamados nodos associativos (3.7 e 13 da Fig. 2), de acordo com a conectividade assumida pelo especialista para a rede neural. Este gráu de ativação de cada associação de neurônio é obtida como uma operação de t-normas de suas saidas. Esta entrada depende da ativação e da relevância dos nodos terminais que chegam. Relevâncias (.5..8,.. na Fig. 2) é associada ao peso da rede sinaptica, e mede a importância de cada peça de informação como suporte para a tomada de decisão. O produto é a operação de intersecção mais comum (t-norma) usada para combinar relevâncio e confiança.

Os nodos de decisão (8 e 14 na Fig. 2) são usados pelo especialista para orientar a navegação das redes neurais. Neurônios de decisão levam a vantagem de

filtrar propriedades de seus terminais para dirigir ao fluxo de informação de nodos de redes diferentes. Por exemplo, os neurônios 8 e 14 da Fig. 2 tanto podem transferir o foco da hipótese para "óleo parado" como proceder a investigação da hipótese "trem de gas" pedindo outras informações.

O CONHECIMENTO CONECCIONISTA

(Leão and Rocha, 1990) introduziram uma técnica de aquisição e representação do tipo de conhecimento coneccionista. Esta técnica foi largamente comprovada por (Machado et al., 1990 e Rocha,1992). A seguir, estão as principais etapas de obtenção do conhecimento do especialista segundo esta técnica:

- é solicitada ao especialista uma lista de problemas do domínio e;
- para cada problema do domínio, é requerido ao especialista uma lista de dados prototípicos de suporte e para ordenar esta lista segundo a sequência temporal, cada peça da informação deve ser obtida em condições reais.

Esta lista ordenada é considerada um conjunto de nodos sensoriais (neurônios 1,2,4,5,6,11 e 12 da Fig. 2) da rede de conhecimento correspondente. Devido a isto é solicitada ao especialista:

- unir os nodos terminais (sensoriais) da rede da mesma forma que ele combina informações de suporte para sua tomada de decisão. Em fazendo isto, o especialista cria o número de neurônios associativos (neurônios 3,7,13 e 8 e 14 da Fig.2) necessário para imitar na prática o "clustering" de dados de suporte, até esses neurônios associativos representarem alternativas possíveis para tomadas de decisões (por exemplo "trem de gas " na Fig.2).

Uma vez descrito o gráfico e obtida a estrutura da rede neural, o especialista:

- deve fornecer o peso de cada arco do grafo. Estes pesos representam a relevância (.5..8 .....na Fig2) e cada peça de informação tem o suporte do raciocínio do especilaista.
- deve prover o conhecimento prototípico usado para "matching" na camada de entrada. Este conhecimento é, em geral, codificado pelas funções de "matching" definindo conceitos tais como alto, baixo, médio, pressão de sucção, temperatura alta e outros (Zadeh, 1975);
- deve descrever as regras (t ou s-normas) usadas para combinar informações nos neurônios de associação/decisão e;
- deve definir os limiares lógicos operacionais a serem desempenhados por cada neurônio associativo. Estas operações descrevem as condições que cada tomada de decisão deve obedecer e o fornecimento das regras para navegar cada rede.

RACIOCINANDO SOBRE O GAS-LIFT E O PROCESSO

Algumas das decisões que devem ser tomadas durante a operação do processo são baseadas nas cartas de gas de superfície e complementadas pelos resultados da produção de óleo. Desde que as cartas de gas de superfície não fornecem todas as informações requeridas para controle do processo e dos poços, muitos outros índices de performance tais como previsões de pressão e temperatura na profundidade de injeção de gas e a vazão de óleo produzido, são usados pelo engenheiro para dar suporte a sua tomada de decisão final. Embora estes índices sejam obtidos como resultados de cálculos definidos usando medidas numéricas de condições da elevaçào artificial, eles são usualmente referidos como expressões qualificadas de alto, médio, pressão baixa, temperatura, vazão, em muitos exemplos.

Estas informações são continuamente obtidas das instalações do processo e dos poços por meio de sensores eletrônicos, que enviam sinal ao módulo escritório central da última variável adquirida ou requerendo ao usuário o fornecimento do mesmo.

Informações como pressão alta de operação, baixo volume de injeção de gas, vazão de produção média, baixa razão gas-líquido, alta temperatura, são usadas aqui tanto como nodos disparadores como evidências complementares para tomada de decisão do especialista. Uma vez ativados os disparadores, um módulo supervisor atua na rede neural correspondente, e outras informações são obtidas dos sensores ou do usuário para suporte da decisão.

VALIDAÇÃO

Para testar a proficiência do SEPLANT, os sintomas resultantes de 35 falhas do processo e dos poços de gas-lift foram apresentadas ao sistema de redes neurais. Durante esta fase, todos os valores de saida foram precisamente classificados dentro de segundos para correção das falhas, resultados estes confirmados pelo uso de dados de campo (cartas de gas de superfície). A tabela 1 mostra a curva média (um dos casos é simulação) para a açào da rede neural em problemas simples.

A tabela 1 lista dados de treinamento e reais de 4 casos envolvidos. As colunas da mesma representam sequencialmente da esquerda para a direita: números médios de nodos de evidências nos grafos de conhecimento, dos casos teóricos (01) e concretos (03) envolvidos.

Tabela 1 Resultados da Simulação

GRAFO	CASO T.	CASO # 1	CASO # 2	CASO # 3
POÇOS DE GAS-LIFT FECHA-DOS	SIM E NÃO	SIM	SIM	SIM
POÇOS SURGÉN-FECHA-DOS	SIM E NÃO	TODOS OU ALGUNS	TODOS OU ALGUNS	TODOS OU ALGUNS
SEPARA-ÇÀO DE O/G/W (NÍVEL %)	45	ATÉ 60	ATÉ 50	ATÉ 50
TRATA-MENTO DE ÓLEO (NÍVEL %)	40	20 NO MÁXIMO	20 NO MÁXIMO	20 NO MÁXIMO
SISTEMA DE BOMBEIO	PARADO	PARADO	PARADO	PARADO

Na série seguinte de simulações. um conjunto de sintomas foi apresentado ao SEPLANT para determinar sua habilidade em executar tarefas generalizadas. Esta capacidade foi examinada introduzindo sintomas de entrada de dois e três malfuncionamentos de poços e processos respectivamente. O caso acima mencionado desperta interesse pois o diagnóstico de falhas múltiplas é difícil de estabelecer usando métodos tradicionais de análise de problemas.

As condições de campo do SEPLANT foram testadas com corridas de vários casos simulados quais sejam um modelo teórico e três casos de campo reais, que validaram a estrutura desenvolvida para construção do SEPLANT.

As redes neurais foram capazes de aprender a associação correta entre evidências e malfuncionamentos de processo e poços. Foram capazes também de diagnosticar múltiplas condições de falhas.

CONCLUSÕES

O SEPLANT simplfica a diagnose de problemas e sumariza relatórios de fácil entendimento. Problemas de superfície (processo e poços) e de fundo (poços) são reconhecidos e importantes parâmetros de produção são calculados.

O SEPLANT minimiza a participação requerida do especialista na diagnose do problema mas não elimina a necessidade do mesmo. O SEPLANT está sendo usado como uma ferramenta de treinamento para o pessoal de projeto e operação. Com este sistema, os usuários podem aprender previamente problemas não familiares do gas-lift (poços e processo) e preverem soluções para estes problemas.

Os resultados obtidos através das simulações dos casos teórico e reais mostraram que a teoria de redes neurais simbólicas codifica e absorve o raciocinio dos especialistas envolvidios na engenharia de análise de sistemas de gas-lift.

A estratégia de operação do SEPLANT pode aumentar a possibilidade de sucessos técnico e econômico da implementação e manuseio inteligentes de poços e processos de gas-lift.

REFERENCES

- Alegre L., Morooka C. K. and Rocha A. F. - Intelligent Approach of Rod Pumping Problems. SPE Petroleum Computer Conference, pg. 249- 255, (1993).

- Brown K. - Artificial Lift Methods - vol. I - USA(1977)

- Garcia G. and Whitman W. W. - Inversion of a Lateral Log Using Neural Networks, SPE Petroleum Computer Conference, pg 295 - 304, (1992)

- Hasan A. R. and Kabir C. S. - Predicting Fluid Temperature Profiles in Gas-Lift Wells, SPE Petroleum Technology Co, pg 673 - 682, (1993)

- Homem F. C. - Offshore Power Generation Reliability and Availability - Msc. Thesis, Cranfield Institute, London (1991)

- Leão B. and Rocha A. F. - Proposed Methodology for Knowledge Acquisition: A Study on Congenital Heart Diseae Diagnosis, Methods Information Medicine - (1990)

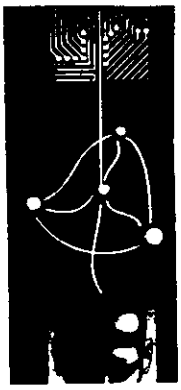
- Machado R. J. and Rocha A.F. - Calculating the Mean knowledge Representation from Mltiple Experts In: Fedrizzi & Kacprzyck. Multiperson Decision Making Models Using Fuzzy Sets and Possibility Theory, Kluwer Academic Plublishers - (1990)

- Patricio A. R. - Sistema Especialista para Apoio à Operação de Plantas Marítimas de Produção - Msc. Thesis, Unicamp - (1992)

- Rocha A. F. - Neural Nets: A Theory for Brains and Machines. Lecture Notes in Artificial Intelligence - Spring Verlag, vol. 638 - (1992)

- Rocha A. F. and Yager R. R. - Neural Nets and Fuzzy Logic, in: Kandel and Langholz (eds), Intelligent Hybrid Systems, CRC Press, pg 13 - 27 (1992)

- Zadeh L. - The concept of A Linguistic Variable and its Application to Approximate Reasoning. Information Sciences, pg 199 - 301 (1975)



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá. 24 a 27 de outubro de 1994

DETERMINAÇAO DA ABERTURA DOS CILINDROS NO PROCESSO DE LAMINAÇÃO, UTILIZANDO REDES NEURONAIS

C.D.M. Pataro†, P. Resende† e H. Helman‡

†□Departamento de engenharia Eletrônica da UFMG, Caixa Postal 1294, Belo Horizonte, MG, 30160-030.
‡ Departamento de engenharia Metalúrgica da UFMG, Caixa Postal 1294, Belo Horizonte, MG, 30160-030.

RESUMO

Neste trabalho é proposto um método para o cálculo da abertura entre cilindros de um laminador, utilizando redes neuronais "backpropagation". O treinamento da rede é desenvolvido a partir de dados obtidos de laminação de amostras do material a ser utilizado, considerando-se as deformações já ocorridas (ou não) e a espessura objetivada, para cada abertura entre os cilindros. Os resultados do método apresentado são comparados àqueles provenientes da utilização de modelos teóricos.

1- INTRODUÇÃO

A laminação consiste na passagem de um material entre dois cilindros que giram de forma a reduzir a espessura do referido material. Por este processo podem ser obtidos produtos planos (chapas, tira, etc) e não planos (barras, cantoneiras,trilhos, etc). Neste trabalho é abordada a laminação a frio de planos, onde os cilindros são lisos e normalmente o cilindro inferior não tem movimento vertical, sendo o ajuste da abertura entre eles, efetuado pelo deslocamento do superior [5].

Quando uma chapa é laminada, há uma deformação elástica em toda a estrutura do laminador, fazendo com que a abertura entre os cilindros seja diferente da espessura objetivada do material para aquele passe. Referindo-se à figura 1, se a rigidez do laminador fosse infinita, a abertura entre os cilindros, g, seria igual à espessura de saída

da chapa, h. Como a estrutura do laminador se deforma, a chapa sai com uma espessura igual a:

$$h = g + s$$

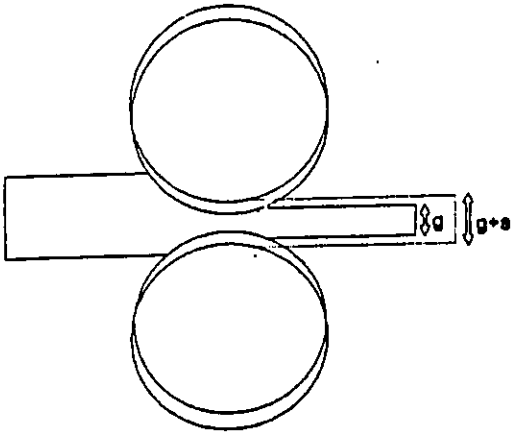


Figura 1 - Deformação da estrutura do laminador.

onde s é a deformação elástica do laminador.

A figura 2 mostra a forma característica da curva carga de laminação x deslocamento da estrutura do laminador. A inclinação da porção linear desta curva é chamada Módulo de Rigidez do Laminador, M. A não linearidade apresentada no início dela é explicada pela eliminação de folgas existentes nos componentes mecânicos e pela redistribuição de cargas. Nos laminadores industriais esta parte não linear é, geralmente, desprezada por não se trabalhar com cargas de laminação desta ordem de grandeza. Assim, a abertura se relaciona com a carga de laminação por meio da seguinte equação [5],[10],[11].

$$g = h - P/M$$

onde g, h,M,P denotam respectivamente a abertura entre cilindros, espessura objetivada, módulo de rigidez do laminador e a carga de laminação

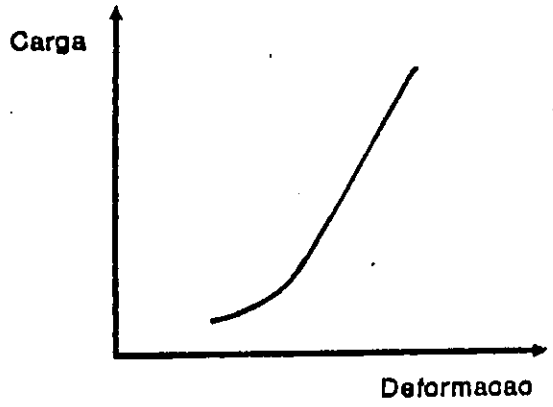


Figura 2 - Variação da deformação do laminador com a carga de laminação.

Na determinação da espessura de saída de uma chapa a ser laminada, é comum a utilização de modelos teóricos para calcular a carga de laminação e , através dela, a abertura entre os cilindros, para cada condição de material e operação. Estas condições envolvem variáveis tais como: curva de fluxo, espessura final objetivada, espessura de entrada da chapa, coeficiente de atrito, carga de laminação, deformação elástica do laminador e recuperação elástica da chapa, dentre outras. Algumas destas condições são de difícil determinação, como o coeficiente de atrito, por exemplo.

Ao se trabalhar com modelos teóricos,consideração especial deve ser dada à deformação dos cilindros, pois devido às elevadas pressões que se desenvolvem na laminação a frio, o raio destes deixa de ser circular , sofrendo uma variação de acordo com a deformação do arco de contato. Esta deformação adquire um perfil consistente com a distribuição de pressões existente na

interface chapa-cilindro[7]. Hitchcock propôs uma solução que considera a distribuição de pressões no arco de contato como elíptica (figura 3) e demonstrou que, após a deformação, o arco de contato adquire um perfil circular, com um raio $R`>R$. Os modelos levam em consideração o raio deformado, R' [5].

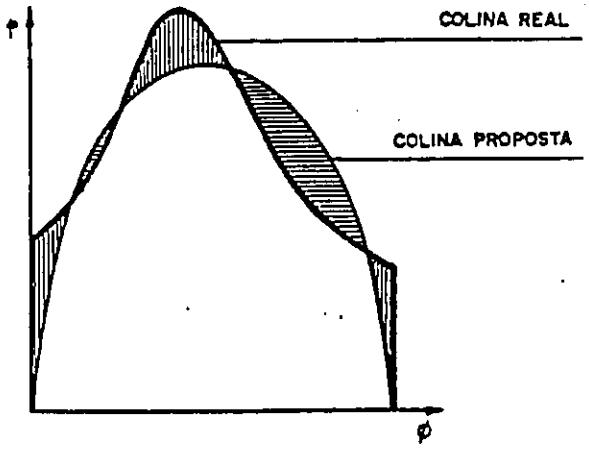


Figura 3 - Distribuição de pressão hipotética atuante nos cilindros.

Além de dados referentes ao comportamento do laminador, os modelos teóricos necessitam ainda de dados a respeito do comportamento mecânico do material. Trata-se da chamada curva de fluxo do material, que tem um perfil semelhante àquele mostrado na figura 4.

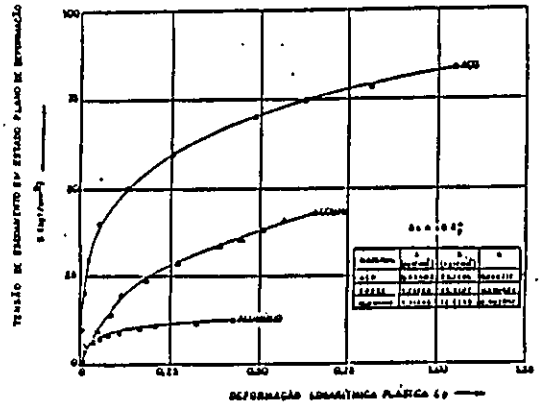


Figura 4 - Perfis típicos de curvas de fluxo.

Pelo que ficou exposto acima, é de fundamental importância o conhecimento da carga de laminação e do módulo de rigidez do laminador para o correto posicionamento dos cilindros. A determinação de M é feita experimentalmente, enquanto a carga de laminação é calculada por meio de modelos.

Atualmente vários trabalhos utilizam redes neuronais "backpropagation" para implementar funções multidimensionais não-lineares[6,8]. O emprego de redes neuronais em laminação tem se mostrado adequado para o cálculo de aberturas entre cilindros de laminadores, levando em consideração a carga calculada por meio de modelos teóricos e as larguras das chapas a laminar [10]. Neste trabalho será feita uma generalização de [10], onde se faz o treinamento de uma rede neuronal de modo que ela aprenda o comportamento de todo o conjunto: laminador, material, condições de laminação, etc.

Uma vez obtidos os pesos, w_{ij}, por meio do treinamento da rede, o tempo de execução dos cálculos para se obter as saídas, o_{pk}, se resume ao tempo de resolução das equações:

$$i_{pj}=f_j^h(net_{pj}^h)$$

$$o_{pk}=f_k^o(net_{pk}^o)$$

onde, neste trabalho, será empregada a função:

$$f(net_i)=\frac{1}{1+\exp(-net_i)}$$

O tempo de resolução destas equações, em relação ao de solução de modelos teóricos, é consideravelmente menor.

2 - MODELOS TEÓRICOS.

Existem diversos modelos teóricos destinados a avaliar esforços relacionados com o processo de laminação de produtos planos[3,9]. Neste trabalho serão feitas experiências para um mesmo material e mesmas condiçoes de laminação, utilizando o modelo de von Karman, ampliado e instrumentado por Alexander[2]

3 - MÉTODO EMPREGANDO REDE NEURONAL

O método empregado utilizou, para treinamento da rede neuronal, seis laminações de duas amostras do material a ser laminado. A Tabela 1 mostra como foram efetuadas estas laminações, de forma a cobrir a faixa de deformações desejada. Estas amostras devem corresponder ao material na forma como este será laminado, ou seja, devem ser da mesma largura, espessura e condição inicial.

Para o esquema de treinamento proposto, onde foi empregado 'backpropagation" [4,6,8,10], as entradas correspondem à deformação logarítmica pré-existente e a espessura de saída da chapa. A saída da rede corresponde à abertura, g, entre os cilindros. Referindo-se à Tabela 1, a primeira chapa é laminada a partir de 0% de deformação, saindo com uma deformação logarítmica de 6%. Novamente a mesma chapa é laminada, ficando então com uma deformação acumulada de 27%, sendo laminada pela terceira vez a partir deste ponto. A segunda chapa sofre uma deformação de 24% no primeiro passe; a partir deste valor ela sofre mais 26% de deformação, sendo laminada pela terceira vez, partindo de 47%. A deformação final deste material amostrado foi de 94%. Embora a carga de laminação seja medida para cada passe, ela não participa do treinamento, pois o que interessa para a rede é determinar a abertura necessária para obter cada espessura objetivada. Empregando os dados mostrados acima, a rede neuronal aprende a calcular a abertura necessária para qualquer valor de espessura objetivada, dentro da faixa de treinamento.

Tabela 1 - Dados empregados no treinamento.

Laminações	Deformação Acumulada (%)	Espessura Obtida (mm)	Abertura (mm)
chapa 1	0	2,83	2,63
chapa 1	06	2,28	1,98
chapa 1	27	1,85	1,54
chapa 2	0	2,35	1,98
chapa 2	24	1,87	1,54
chapa 2	47	1,14	0,67

A carga de laminação não participa do treinamento da rede neuronal, mas deve ser levada em consideração na fase de aprendizagem, a fim de não exceder a capacidade do laminador. Assim, por meio das cargas obtidas em cada laminação é feita uma previsão aproximada da próxima carga . Para tanto, pode-se empregar um método de extrapolação. Neste trabalho foi empregado o método de Richardson [1], que se mostrou apropriado. Caso não exista nenhuma limitação quanto à carga de laminação na faixa de deformações para a qual se pretende treinar o sistema, esta providência é dispensável.

4 - RESULTADOS OBTIDOS.

4.1 - Modelo teórico.

Como material para exemplo, utilizou-se alumínio. De acordo com os resultados dos cálculos de carga de laminação obtidos através do modelo teórico e do módulo de rigidez do laminador empregado, foram feitas

algumas laminações com o material, medindo-se, para cada espessura objetivada, a espessura de saída da chapa. Estes valores são apresentados na Tabela 2.

Tabela 2 -Resultado de laminações para o modelo teórico.

Abertura Calculada (mm)	Espessura Objetivada (mm)	Espessura Obtida (mm)	Erro (%)
2,37	2,8	2,69	4,1
2,23	2,56	2,48	3,2
1,37	2,0	1,8	1,1
1,1	1,6	1,45	10,3

4.2 - Redes neuronais.

As laminações descritas na Tabela 1 correspondem a amostras retiradas do mesmo material empregado para o experimento com modelo teórico. Na Tabela 3 podem ser vistos os resultados de algumas laminações utilizando a rede para calcular a abertura entre cilindros em situações dentro da faixa de treinamento.

Tabela 3 - Resultados de laminações após treinamento.

Abertura Calculada (mm)	Espessura Objetivada (mm)	Espessura Obtida (mm)	Erro (%)
2,6	2,8	2,78	0,7
2,4	2,6	2,58	0,8
1,8	2,0	2,07	3,4
1,12	1,6	1,58	1,0

6 - CONCLUSÕES.

O método utilizando redes neuronais se mostrou adequado para o cálculo de aberturas no processo de laminação, tornando o procedimento independentes do conhecimento de variáveis mecânicas e metalúrgicas do processo tais como atrito, módulo de rigidez do laminador, curva de fluxo do material, etc. A aplicação de redes neuronais se mostrou também eficiente no que se refere aos tempos de cálculo, que são reduzidos em relação aos tempos gastos com a utilização de modelos teóricos.

7 - REFERÊNCIAS BIBLIOGRÁFICAS.

[1] - ACTON, F. Numerical Methods that Work., New York:Harper & How, 19970

[2]- ALEXANDER, J.M. On the Theory of Rolling. Proc. R.Soc. Lond. A.326, 535-563 (1972)

[3] - BLAND, D.R. and FORD,H., "The Calculation of Roll Force and Torque in Cold Strip Rolling with Tensions", Proc. Instn. Engrs., Vol. 159, pp. 144-153. 1948.

[4] - FREEMAN, J.A. & SKAPURA, D.M. Neural Networks. Algorithms, Applications, and Techniques. Addison-Wesley Publishing Company, 1991.

[5] - HELMAN, H. & CETLIN, P.R. Fundamentos da Conformação Mecânica dos Metais. Guanabara Dois. Rio de Janeiro. 1983.

[6] - HUNT, K.J.; SBARBAR, D.; ZBIKOWSK, R. and GAWTHROP, P.J."Neural Networks for Systems"- A Survey. Automática, Vol. 28, no 6, pp. 1083 - 1112, 1992.

[7] - JORTNER, D.;OSTERE, J.F. and ZOROWSKI, C.F., Än Analysis of Cold Strip Rolling", Int. J. Mech. Sci., Vol. 2, pp. 179-194,1960.

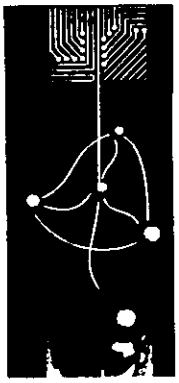
[8] - OLIVEIRA, J.B., COTRIM, D.C. & ALVES, J.C., "Redes Neuronais - Backpropagation e Kohonen- Fundamentos e Aplicações", III Seminário de Automação na Indústria, Volta Redonda, RJ, pp. 1-18, 1993.

[9] - OROWAN E., "The Calculation of Roll Pressure in Hot and Cold Flat Rolling" , Inst. Mech. Engrs. (London) (Journal and Proccedings), Vol. 150, pp. 140-167, London, 1942.

[10] - SCHMIDT, W. G.; PATARO, C.D.M.; RESENDE, P. & HELMAN, H., "Posicionamento Automático de Cilindros em Laminadores Empregando Treinamento com Redes Neuronais", III Seminário de Automação na Indústria, Volta Redonda, RJ, pp. 19-26, 1993.

[11] - SUSUKI, H. & ATAKA, M. "Study on Rolling Mill Moluli, Effets os Rolling Mill Moduli, and Optimum Arrangement of Mill Moduli for Tandem Strip Mill". Report of the Institute of Industrial Science the University of Tokyo, Vol. 25. número 1, December, 1975.

Anotações

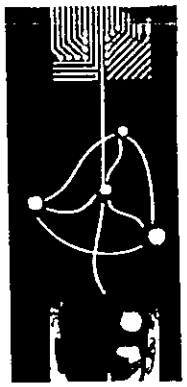


1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

SISTEMAS ELÉTRICOS DE POTÊNCIA

Anotações



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

APPLYING ASSOCIATIVE MEMORIES TO FAULT LOCATION IDENTIFICATION

A.P. Alves da Silva[1] A.H.F. Insfran[2]
P.M. da Silveira[1] G. Lambert-Torres[1]

1. Escola Federal de Engenharia de Itajubá - Brazil
2. Itaipú Binacional - Brazil/Paraguay

Abstract: Faults producing load disconnections or emergency situations have to be localized as soon as possible to start the electric network reconfiguration, restoring normal energy supply. This paper proposes the use of artificial neural networks (ANNs), of the associative memory type, to solve the fault localization problem. The main idea is to store measurement sets representing the normal behavior of the protection system into associative memories. Afterwards, these memories are employed on-line for fault location estimation from the protection system equipment status. The associative memories work correctly even in case of malfunction of the protection system. Although the ANNs are trained with single contingencies only, their generalization capability allows a good performance for multiple contingencies.

1 Introduction

In a power system substation, faults that produce load disconnections or emergency situations have to be localized as soon as possible. Fault localization is necessary to start the substation reconfiguration for restoring normal energy supply. However, the identification of the faulted points is not always an easy task, delaying the restoration procedures. This usually occurs when the protection system does not behave as expected.

Substations in commissioning phase or even the ones already in operation, but with complex constructive and operational natures, can have high indices of protection system failure. In these substations, fault localization can take a long time due to the great amount of information to be analyzed. Even visual inspection can be required.

The difficulty in identifying the faulted points significantly increases in non-conventional substations, as gas-insulated (GIS) ones [1].

This paper proposes the use of artificial neural networks (ANNs), of the associative memory type, to solve the fault localization problem in substations. The idea is to store, into associative memories, measurement sets representing circuit breaker and relay status (normal behavior) corresponding to possible single faults. Afterwards, these associative memories are used on-line to estimate fault locations (equipment, phase and compartment), even in case of misoperation of the protection system. Although the ANNs are trained with single contingencies only (stored cases), their generalization capability allows a good performance for multiple (simultaneous) faults.

2 Associative Memory ANNs

Intelligent systems have been successfully applied to the problem of fault diagnosis. Two approaches have been used to solve this problem: symbolic expert systems [2,3] and neural networks [4]. Expert systems have been criticized for requiring a great effort to build (knowledge acquisition) and maintain the knowledge base.

ANNs offer a simple and more robust solution to the fault diagnosis problem due to their noise suppression capacity, training power and adaptability. The noise suppression ability allows them to correctly localize faults, even in case of protection system misoperation. The capability of training using samples of solved cases (supervised training) reduces the development time very much. Finally, the ANNs adaptability makes the maintenance job trivial.

The proposed approach is an innovation compared with reference [4] where the ANNs are used as pattern recognizers, i.e., simple models are desirable. In [4], the ANNs training process is based on normal and abnormal operational conditions of the protection system, and not on the protection philosophy. As the numerous possibilities are not well defined, the limited number of

cases used for training hardly produces good generalization.

The new approach utilizes the ANNs as associative memories, where models with redundancy are desirable. The novelty is related to the difference between the learning (pattern recognizers) and memorization (associative memories) concepts. The second concept is more useful when the population of interest is well defined.

The general characteristics required from an associative memory include the following abilities [5]:
- to store many associated stimulus/response signal pairs;
- to accomplish this storage through a self-organizing process;
- to store this information in a distributed, robust (highly redundant) manner;
- to generate the appropriate response signal on receipt of the associated stimulus signal;
- to regenerate the correct response signal although the input stimulus signal is distorted or incomplete; and
- to add to existing memory.

Associative memory models can be divided into two categories [6,7]:
1) information processing models, which employ programs for testing, comparing, analyzing, manipulating and storing information; and
2) ANN models, which implement the basic functions of a selective associative memory using a collection of relatively simple elements connected to one another.

Optimal Associative Memories

The optimal nonlinear associative memory (ONAM), an ANN paradigm introduced by Kohonen [8], is selected because of its interpolative response, its least-squares storage degradation, and its well-understood mapping. The ANN models for associative storage based on feedback networks [9-11] are not suitable for the fault localization problem. In addition to the low storage capacity, their nearest-neighbor response (besides the difficulty to control the "attractors", which can generate responses not related to the nearest-neighbor) does not allow the localization of simultaneous faults without explicitly storing them. The interpolative response of an optimal associative memory allows that.

The purpose of an optimal associative memory is to obtain optimal transformations such that, with respect to a wanted input-output transfer relation, the effect of noise or other imperfections on the input signals would be minimized. As the ONAM is a simple extension of the optimal linear associative memory, the latter is described first.

The Optimal Linear Associative Memory

Let $\underline{x}_1=(x_1,x_2, \ldots, x_m)^t$, $\underline{x}_2, \ldots, \underline{x}_r$ be an ensemble of input signals (status of circuit breakers and protection relays) in a representation space $\Re^m$, and $\underline{y}_1=(y_1,y_2, \ldots, y_n)^t$, $\underline{y}_2, \ldots, \underline{y}_r$ their associated output signals (fault location) in $\Re^n$. The signals are assumed to be linearly transformed by the following transfer relation (recall procedure):

$$Y = X.W \qquad (1)$$

where $X = [\underline{x}_1\ \underline{x}_2\ \ldots\ \underline{x}_r]^t$, $Y = [\underline{y}_1\ \underline{y}_2\ \ldots\ \underline{y}_r]^t$ and W is an m x n matrix. The components of W, w_{ij}'s, represent the interconnection weights leading from element "i" to element "j" in the corresponding two-layer feed-forward ANN (Figure 1).

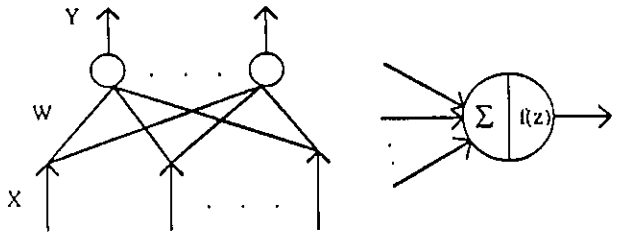


Figure 1 - ANN of the optimal associative memory type, where f(z) = z.

The following analysis is developed considering an autoassociative recall (Y=X). The vectors $\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_r \in \Re^m$ span a subspace $L \subset \Re^m$. An arbitrary vector $\underline{x} \in \Re^m$ is uniquely described as the sum of two vectors $\hat{\underline{x}} \in \underline{x}$, i.e., the orthogonal projections of $\underline{x}$ on L and on the orthogonal complement of L, respectively. Therefore, $\hat{\underline{x}}$ is the best linear combination of the input signals $\underline{x}_k$(k=1, .., r) that approximates $\underline{x}$ in the sense of least-squares

Orthogonal projection operations have the property of correcting noisy and/or incomplete input signals towards the stored ones. If an input signal is a noisy version of one of the stored signals $\underline{x}_k$, $\underline{x} = \underline{x}_k + \underline{\varepsilon}$, where $\underline{\varepsilon}$ is a random error, then in general, $\hat{\underline{x}}$ is a better approximation of $\underline{x}_k$. It can be shown that for the case in which $\underline{\varepsilon}$ has a symmetrical multivariable Gaussian radial distribution in $\Re^m$, its orthogonal projection on L, $\hat{\underline{\varepsilon}}$, has a distribution with the following standard deviation:

$$\delta(\|\hat{\underline{\varepsilon}}\|_2 = \|\hat{\underline{x}} - \underline{x}_k\|_2) = (r/m)^{1/2}.\|\underline{x} - \underline{x}_k\|_2 \qquad (2)$$

The input signal noise is attenuated by the orthogonal projection if $r < m$. Although the analysis above is related to an autoassociative memory, the same noise attenuation factor, $(r/m)^{1/2}$, can be applied to the output vectors of a heteroassociative memory (Y ≠ X) [8].

With regard to Eq.1, the optimal least-squares correlation of X and Y (training procedure) is defined as

$$W = X^{\dagger} . Y \quad (3)$$

where

i) $X^{\dagger} = X^{-1}$ (4)

if m = r and X is nonsingular,

ii) $X^{\dagger} = X^{t} (X X^{t})^{-1}$ (5)

if the rank of X is equal to r, i.e., the input signals to be memorized are linearly independent (r < m), and

iii) $X^{\dagger} = (X^{t} X)^{-1} X^{t}$ (6)

if the rank of X is equal to m, i.e., the input channels are linearly independent (m < r).

When exact solutions to Eq.3 exist, i.e., Eqs. 4 and 5, then W is the particular solution that supplies the associative mapping with the best error tolerance. When exact solutions do not exist, Eq.6, W is the best approximate solution in the least-squares sense.

With this encoding scheme, the strong condition of orthogonality among stored signals is not required to get perfect recall (when an exact solution is possible), as would be the case for an encoding scheme based on the Hebb's rule [12].

The Optimal Nonlinear Associative Memory

For linear associative mappings, a necessary condition for an exact solution to exist is the linear independence of the signals to be memorized (rows of $X(\underline{x}_k^{t}$'s)). When the number of signals is greater than their dimensionality, this condition is necessarily violated and no exact solution exists. It is possible to overcome this limitation by using nonlinear transformations to enhance the original input signal representation. With a nonlinear preprocessing transformation, the dimensionality of the $\underline{x}_k$'s is increased; consequently, the probability for the transformed vectors to become linearly independent (distinguishable) also increases. Another desirable effect produced by an enlargement of the dimensionality of the signals is the improvement of the noise attenuation factor $(r/m)^{1/2}$.

The selected nonlinear transformation is a polynomial one. Polynomial transforms belong to a class of least-squares problems that are linear in the parameters and nonlinear with respect to the input vectors. The chosen polynomial transform is as follow:

$$\underline{x}_k = (x_1, x_2, \ldots, x_m)^t \rightarrow$$

$$\rightarrow \underline{p}_k = (x_1, x_2, \ldots, x_m, x_1x_2, x_1x_3, \ldots, x_{m-1}x_m, x_1x_2x_3, \ldots, x_{m-2}x_{m-1}x_m)^t \quad (7)$$

The name ONAM does not imply that the nonlinear transformation is optimal in an absolute sense. The optimality criterium is applied to estimate the parameters, given a certain nonlinear transformation.

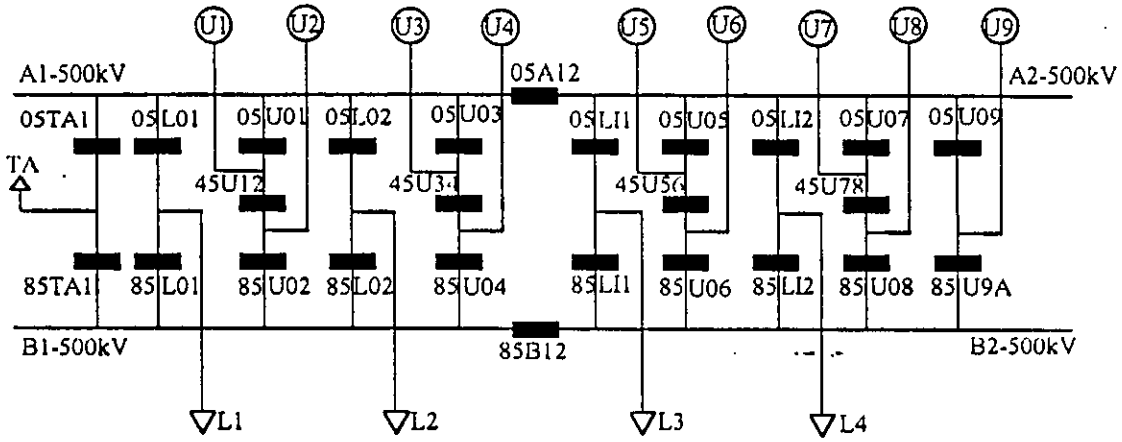


Figure 2 - 500 kV Gas-Insulated Substation.

3 The Proposed Approach

The proposed scheme was idealized for the 500 kV, 50 Hz, Itaipú gas-insulated substation, which used to present difficulties in fault localization due to its large number of compartments (321). This substation is represented in Figure 2, where there are 9 generation units, 4 transmission lines, 4 buses, 1 auxiliary transformer and 26 circuit breakers. The protection system has 66 relays. There are 18 generator differential relays, 8 bus differential relays, 2 transformer differential relays, 8 distance relays, and 30 circuit breaker failure relays.

The status of circuit breakers and protection relays can be used to localize faults in a substation. The status of these equipment can be represented by binary signals in which the "0" characterizes a closed circuit breaker or a non-operating relay, while "1" characterizes an open/tripped circuit breaker or an operating relay. A typical training set is presented in Table 1. The 26 first input channels represent the circuit breaker status, while

the other 40 channels represent the relay status. For example, in case of a fault in bus B2, the value "1" for channels 14, 22, 23, 24, 25 and 26 shows that the circuit breakers 85B12, 85LI1, 85U06, 85LI2, 85U08 and 85U09A have opened. The value "1" for channels 33, 34 and 38 shows that the relays 87B2/P, 87B2/A and BF/B2 have operated. The value "0" for the other channels represents closed circuit breakers and relays that have not operated. The output channels indicate a single fault in bus B2. The number of output channels is the same as the number of equipment. In this example, the output signal generated by the ANN that identifies the faulted equipment activates the ANN responsible for bus B2, which is utilized to localize the faulted phase and compartment.

In this way, the fault localization problem is decomposed as shown in Figure 3. Based on the states of relays and circuit breakers, 66 and 92 input binary channels feed the EQUIP and CB's ANNs, respectively. CB's is responsible for detecting defective circuit breakers. EQUIP and CB's form the ANNs main group. These ANNs input channels contain information about which equipment are defective and the post-fault substation topology. No information is supplied regarding the faulted phase and compartment. A second group of ANNs is used for that. The input channels for this new group contain information of distance relays situated in the GIS (24 = 4 lines x 3 phases x 2 (double primary protection system)) and in a neighbor substation (12 = 2 lines x 3 phases x 2), of bus differential relays of the GIS (12 = 4 x 3), and of pressure drop relays (107 = 321 / 3; only one alarm for the three phases) also situated in the GIS. Therefore, there are 155 inputs to the second group of ANNs. The number of input channels for each of these ANNs, and the number of ANNs for each type of equipment, including circuit breakers, are shown in Figure 3.

All the ANNs are trained considering normal operation of the primary protection system. The basic idea behind the problem decomposition is to reduce the number of cases that should be memorized if a single ANN were employed. With the reduction on the number of cases to be memorized by each ANN of the proposed approach, smaller ANN architectures can be used for the same noise attenuation factor (significantly reducing the training time).

4 Tests

The ANNs performance is verified using 21 historical cases. There are 17 single contingencies and 4 multiple contingencies. Relay and/or circuit breaker misoperations occur in the 21 faults. All historical cases are correctly solved by the ONAMs (Kohonen nets).

Figure 4 is related to the behavior of the EQUIP ANN with respect to the number of auxiliary input variables (generated by the nonlinear transformation defined by Eq.7). The graph vertical axis shows the difference between the largest and the second largest value of the output channels of the trained ANN. The reliability of a certain output signal is as large as this difference is closer to 1. Figure 4 is built based on the average result for 6 (randomly chosen) of the 17 single faults mentioned before. The best performance happens when the number of auxiliary input variables is equal to 2000. From now on, all the results from EQUIP are obtained using 66 original input channels plus 2000 auxiliary channels (m=2066).

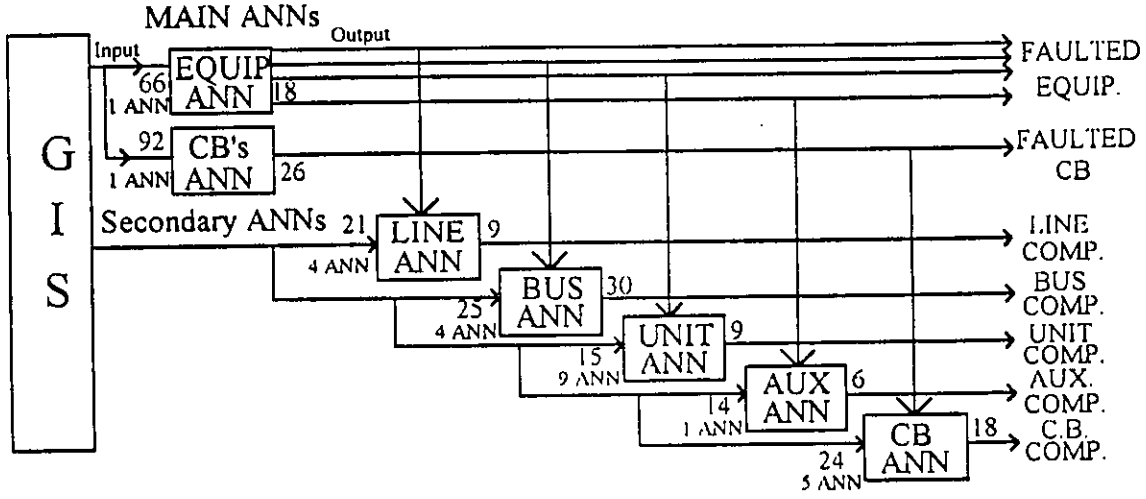


Figure 3 - Decomposition of the Fault Localization Problem.

Table 1 - Training Set for Identifying Faulted Equipment.

	Input	FAULT LOCATION																		
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
			A1	A2	B1	B2	L1	L2	L3	L4	U1	U2	U3	U4	U5	U6	U7	U8	U9	A1
C	05TA1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
.	05L01	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
B	05U01	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
.	05L02	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	05U03	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	05A12	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	45U12	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
	45U34	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
	85TA1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	85L01	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	85U02	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	85L02	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	85U04	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	85B12	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	05LI1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	05U05	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	05L12	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	05U07	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	05U09	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	45U56	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
	45U78	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
	85LI1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	85U06	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	85LI2	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	85U08	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	85U09A	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0
R	87A1/P	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	87A1/A	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	87B1/P	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	87B1/A	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Y	87A2/P	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	87A2/A	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	87B2/P	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	87B2/A	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BF/A1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BF/A2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BF/B1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	BF/B2	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	21P/L1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	21A/L1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	21P/L2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	21A/L2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	21P/L3	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	21A/L3	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	21P/L4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	21A/L4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	87V/U1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	87TR/U1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	87V/U02	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	87TR/U2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	87V/U03	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	87TR/U3	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	87V/U04	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	87TR/U4	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	87V/U05	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	87TR/U5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	87V/U06	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	87TR/U6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	87V/U07	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	87TR/U7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	87V/U08	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	87TR/U8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	87V/U09	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	87TR/U9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	87	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	87TA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

	Output	FAULT LOCATION																		
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
			A1	A2	B1	B2	L1	L2	L3	L4	U1	U2	U3	U4	U5	U6	U7	U8	U9	A1
E	BUS A1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	BUS A2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
U	BUS B1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	BUS B2	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	LINE 1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
.	LINE 2	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	LINE 3	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	LINE 4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	UNIT 1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	UNIT 2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	UNIT 3	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	UNIT 4	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	UNIT 5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	UNIT 6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	UNIT 7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	UNIT 8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	UNIT 9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	AUX.TR.	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

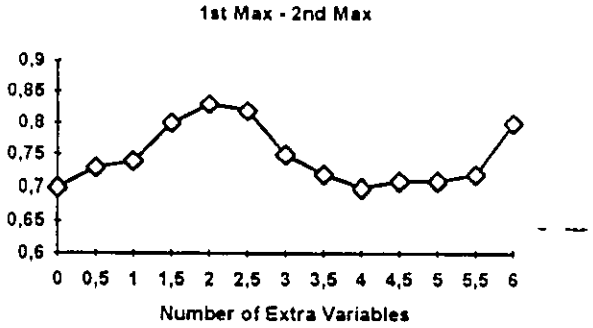


Figure 4 - EQUIP Performance with respect to the Number of Input Variables.

For comparison, Figures 5(a) and 5(b) present the performance of an associative memory based on the Hamming distance with the same objective of EQUIP. The faults are identified by the smallest Hamming distances (nearest-neighbors) between the vector representing the current states of circuit breakers and relays, and the 19 stored vectors. There is one stored vector for each of the 18 equipment of interest, plus one vector associated to the condition of non activation of the protection system.

The vertical axes of the graphics show the Hamming distances from historical contingencies to each stored case, represented in the horizontal axes. In Figure 5(a), curves related to 4 single contingencies (fault in bus 2, fault in line 3, fault in line 4, and fault in generator 8) are shown. The minimum Hamming distance correctly indicates the faulted equipment for the 4 cases, and consequently the number of relays and circuit breakers, among those that serve as input to EQUIP, that have not operated as expected.

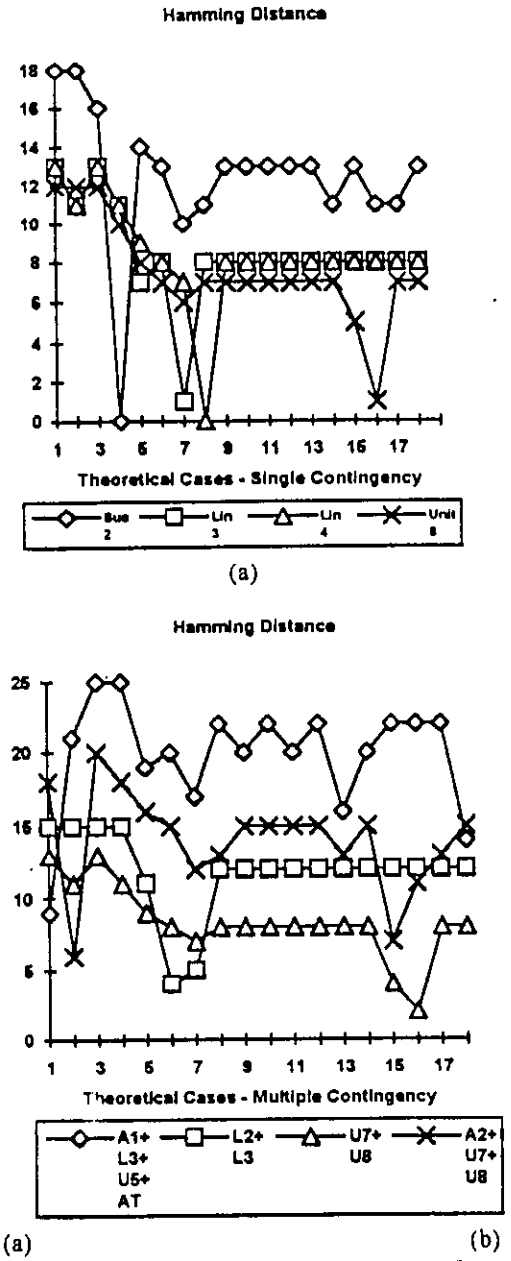


Figure 5 - Associative Memory based on the Hamming Distance

A graph analogous to 5(a) is displayed in Figure 5(b). This time, the curves are related to multiple contingencies. Once more, the l smallest Hamming distances, where l is the contingency order are correctly associated to the faulted equipment. For example, for the multiple contingency A2+U7+U8 the points 2, 15 and 16 represent the corresponding equipment. However, the difference between the forth and the third smallest Hamming distance does not appropriately characterize the contingency order (previously unknown). Analogously, in Figure 5(a) the difference between the second smallest and the minimum Hamming distance does not characterize single faults. For example, the single fault in the generation unit 8 (Figure 5(b)) has value 4 for this difference, while the multiple fault A1+L3+U5+AT has value 5 for the same difference (second smallest minus minimum).

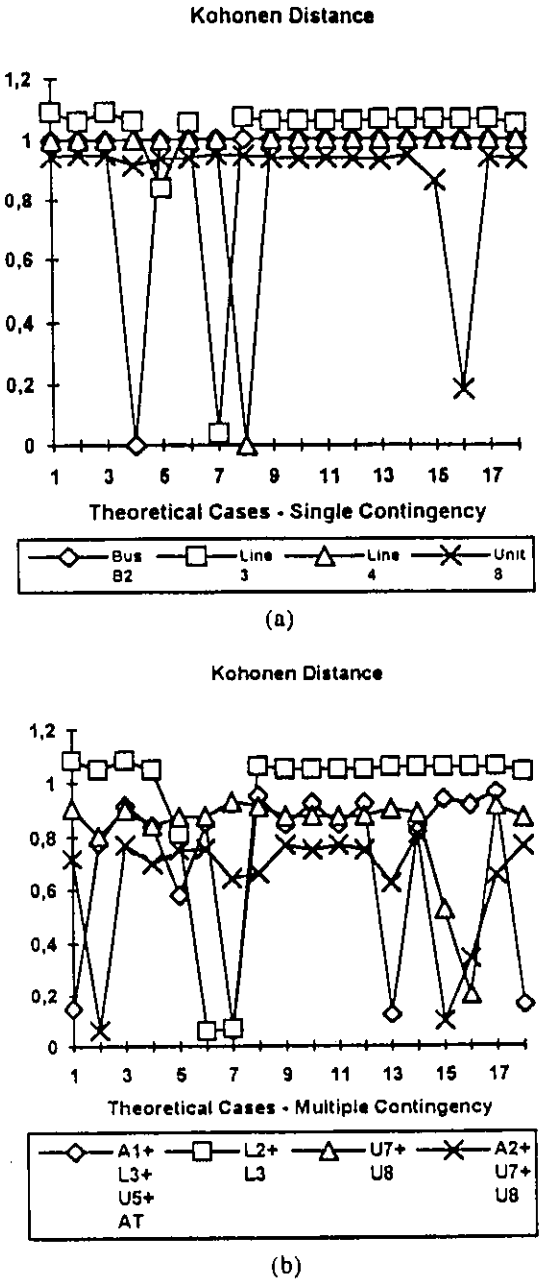


Figure 6 - Kohonen Net (ONAM) Results.

The same conclusions are reached with the Euclidean distance instead of the Hamming distance. Figures 6(a) and 6(b) show graphics analogous to the ones presented in Figures 5(a) and 5(b), employing the Kohonen net this time.

It can be observed that the contingency order is much better defined by the Kohonen net. This becomes clear in Figure 7, where the dispersion that characterizes the contingency order is displayed in a normalized scale for 6 randomly selected historical cases. As the dispersion value 1 is the best characterization of the contingency order, it is clear that the Kohonen net has a superior performance compared with the nearest-neighbor associative memories based on the Hamming and Euclidean distances.

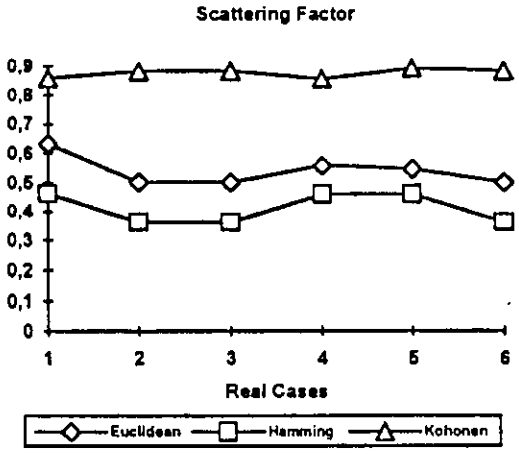


Figure 7 - Dispersion Factor for the Three Associative Memories.

Finally, Figure 8 exhibits results from one of the ANNs belonging to the second group. After EQUIP has identified bus B1 as the defective equipment, the fault location, i.e. the defective compartment, is indicated by the ANN responsible for B1 (bus B1 has 30 compartments; 10 compartments/phase). Once more the results are excellent. The training process of each ANN takes a few seconds in a 486-DX2 66 MHz microcomputer.

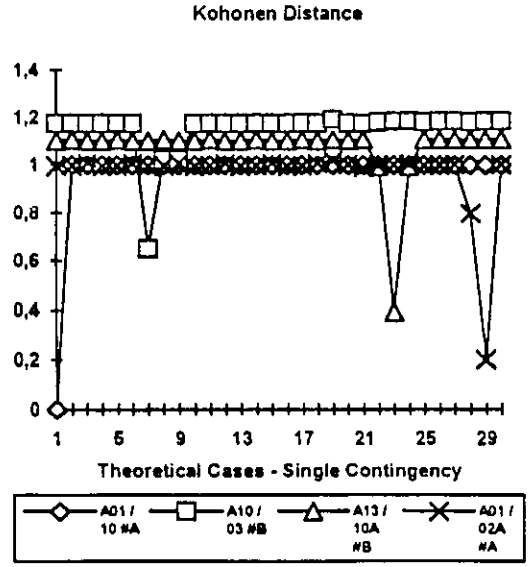


Figure 8 - Results of the ANN responsible for Bus B1.

5 Conclusions

A new approach to the fault localization problem is introduced. In a previous project the authors participated in the development of an expert system to solve the same problem for the Itaipú system [13]. The advantages of the ANN approach are the following:

- The development time of the fault localization system based on ANNs is about 10 times less than one required by the expert system approach. The expert system maintenance takes more time also, because of the necessity of knowledge base consistency checking.

- In general, the great disadvantages of an ANN compared with an expert system is the mapping "opacity" of the ANN. However, for the fault localization problem, the expert system explanation capability does not supply relevant information that cannot be obtained from ANNs, too. This is because solution justification is more useful when the protection system does not behave as expected. In these situations, besides fault localization, it is important that the operator be aware of which relays and/or circuit breakers have not correctly operated. This information can also be obtained via ANNs, comparing the current input signal with the stored input signal corresponding to the current output signal.

- Nevertheless, the ANNs have been more robust than the expert system in situations of protection system misoperation.

Although the proposed approach has been used for a GIS, it can be applied to power system operation centers.

ACKNOWLEDGEMENTS

This work was supported by Itaipú Binacional and the Brazilian National Research Council (CNPq).

REFERENCES

[1] R. Fragiacomo, R. Pani, and A. Tomassi: "Problems Relating to Protection of HV Metal-Enclosed Gas-Insulated Substations (GIS)", Second International Conference on Developments in Power-System Protection, London, June 1980, pp. 6-9.

[2] C. Fukui and J. Kawakami: "An Expert System for Fault Section Estimation using Information from Protective Relays and Circuit Breakers", IEEE Transactions on Power Delivery, Vol. PWRD-1, No. 4, October 1986, pp. 83-90.

[3] T. Kimura, S. Nishimatsu, Y. Ueki, and Y. Fukuyama: "Development of an Expert System for Estimating Fault Section in Control Center based on Protective System Simulation", IEEE Transactions on Power Delivery, Vol. 7, No. 1, January 1992, pp. 167-172.

[4] K.S. Swarup and H.S. Chandrasekharaiah: "Fault Detection and Diagnosis of Power Systems using Artificial Neural Networks", First International Forum on Applications of Neural Networks to Power Systems, Seattle, July 1991, pp. 102-106.

[5] Y.-H. Pao: *Adaptive Pattern Recognition and Neural Network*, Addison Wesley, 1989.

[6] A.P. Alves da Silva, V.H. Quintana, and G..K.H. Pang: "Associative Memory Modelsfor Data Processing", International Journal of Electrical Power & Energy Systems, Vol. 14, No. 1, February 1992, pp. 23-32.

[7] A.P. Alves da Silva, V.H. Quintana, and G.K.H. Pang: "A Probabilistic Associative Memory and Its Application to Signal Processing in Electrical Power Systems", Engineering Applications of Artificial Intelligence Journal, Vol. 5, No. 4, 1992, pp. 309-318.

[8] T. Kohonen: *Self-Organization and Associative Memory*, Springer-Verlag, 1989.

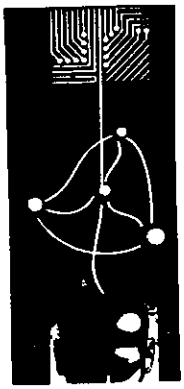
[9] J.J. Hopfield: "Neurons with Graded Response have Collective Computational Properties like those of Two-State Neurons", Proceedings of the National Academy of Sciences (USA), Vol. 81, May 1984, pp. 3088-3092.

[10] B.Kosko: "Bidirectional Associative Memories", IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-18, No. 1, January/February 1988, pp.49-60.

[11] A. Atiya and Y. Abu-Mostafa: "A Method for the Associative Storage of Analog Vectors", in *Advances in Neural Information Processing Systems 2*, D.S. Touretzky (Ed.), Morgan Kaufmann, 1990, pp. 590-595.

[12] J. Hertz, A. Krogh, and R.G. Palmer: *Introduction to the Theory of Neural Computation*, Addison-Wesley, 1991.

[13] R. de Lepeleire, S.M. Francsak, F.M. Vargas, R.D. Siqueira, W.L. Pagliuca, e G. Lambert Torres: "Study of the Use of an Expert System for Post-Fault Disturbance Analysis in Substations", V ERLAC - CIGRÉ Latin-American Meeting, Cidade del Este, Paraguay, May 1993 (in Portuguese).



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

Diagnose em Sistemas de Potência utilizando Redes Neurais

Victor Navarro A. L. da Silva[1]
navarro@fund.cepel.br

Guilherme Nelson F. de Souza[1]
gnelson@fund.cepel.br

Resumo : O principal objetivo deste trabalho é estudar a aplicação de Redes Neurais para diagnose em Sistemas de Potência auxiliando o operador a determinar os motivos que originaram estes distúrbios.
Para tal, será apresentada em detalhes uma análise do treinamento e simulação em um sistema elétrico simplificado. Este protótipo é capaz de diagnosticar não somente falhas simples mas também falhas múltiplas. Além de ser apto a diagnosticar falhas mesmo quando o conjunto de alarmes que a caracterizam estiver incompleto.
Os resultados obtidos demonstram que a Rede Neural é uma técnica extremamente viável e promissora para solução deste e de outros problemas semelhantes existentes no setor elétrico.

I - Introdução:

Os Sistemas Elétricos de Potência vêm passando por um grande desenvolvimento nas últimas décadas, procurando melhorar cada vez mais o processo de produção, transmissão e entrega de energia elétrica.

Esta evolução acarretou um aumento crescente da complexidade dos Sistemas Elétricos, exigindo o desenvolvimento de sistemas de supervisão, controle e automação cada vez mais sofisticados com o objetivo de garantir um funcionamento seguro e confiável.

Entretanto os Sistemas Elétricos continuam sujeitos a distúrbios e normalmente quando eles ocorrem, os operadores destes sistemas tentam solucioná-los desenvolvendo três atividades básicas: ***detecção dos problemas existentes, diagnóstico e determinação das ações corretivas***.

Na detecção dos problemas existentes, os operadores contam com o auxílio do sistema de supervisão que lhes informa todos os alarmes gerados. Normalmente esta quantidade de alarmes é elevada dificultando enormemente o trabalho de identificação das falhas que podem ter gerado tais alarmes (diagnóstico).

Em relação ao diagnóstico, os operadores dependem quase que exclusivamente das suas experiências e conhecimentos e precisam varrer toda a lista de alarmes gerada para

(1) CEPEL - Centro de Pesquisas de Energia Elétrica
Caixa Postal: 2754 CEP: 20.001-970
Rio de Janeiro - Brasil
Tel: 598-2134 Fax: 260- 1340 / 260 - 6211

poder chega a uma explicação razoável sobre as causas dos distúrbios (falhas).

Por último, as ações corretivas são determinadas a partir do diagnóstico alcançado anteriormente e mais uma vez, da experiência e conhecimento do especialista.

Assim, vários fatores podem acabar levando os operadores a fazerem um diagnóstico errado, como a quantidade excessiva de alarmes, a fadiga, o estresse, a falta de experiência, a dificuldade natural em lidar com situações não usuais entre outros. Este diagnóstico falho acaba retardando a tomada das ações corretivas apropriadas, prejudicando a segurança e eficiência de operação do sistema.

Devido a estes motivos, tem-se notado um interesse cada vez maior no desenvolvimento de sistemas que possam ajudar o operador nas etapas de diagnóstico e determinação das ações corretivas [Wollenberg 86], [Tanaka *et. al.* 89], [Kim *et. al.* 91], [Rodriguez 92].

Este trabalho pretende analisar a utilização de Redes Neurais Artificiais na solução deste problema e com isso, contribuir para melhorar o desempenho e reduzir o estresse do operador dando-lhe mais segurança e agilidade no processo de tomada de decisão sobre a causa do distúrbio.

Na próxima seção, faremos uma breve introdução à Redes Neurais. Na seção III, mostraremos como aplicar as Redes Neurais na Diagnose de Sistemas de Potência. Para tal, será apresentada em detalhes uma análise do treinamento e simulação em um sistema elétrico simplificado de uma usina. Finalmente na seção IV, apresentaremos algumas conclusões, sugerindo algumas melhorias e pesquisas futuras.

II - Redes Neurais Artificiais:

O *Cérebro* humano possui dezenas de bilhões de neurônios densamente interconectados (*Redes Neuronais*)[1] , que demonstram grande capacidade para armazenar e processar informações.

O conhecimento sobre o funcionamento de um neurônio, infelizmente, ainda não é o bastante para explicar tarefas extremamente complexas (tais como entender, lembrar, memorizar, generalizar entre outras) que o cérebro é capaz de realizar. Entretanto, o que deve ser destacado é que o cérebro realiza todas essas tarefas de forma eficaz, apesar dos neurônios serem sensivelmente mais lentos que os processadores usados nos computadores atualmente. Possivelmente, toda essa eficiência não está na capacidade individual de um neurônio mas sim no conjunto formado pela quantidade muito grande deles (dezenas de bilhões) e pela forma como eles estão conectados.

Estas características acabaram motivando os cientistas a tentar imitar o modo de funcionamento do cérebro, procurando reproduzir artificialmente as redes neuronais biológicas e usá-las como ferramenta de computação em diversas áreas.

Uma Rede Neural Artificial consiste de vários neurônios (denominados de elementos processadores) organizados em Camadas. Uma rede típica possui uma seqüência de camadas interligadas totalmente ou aleatoriamente com as camadas adjacentes.

Existem três tipos de camadas: Entrada, Saída e Escondida. As camadas de entrada e saída são respectivamente aquelas de apresentação das informações da rede e de visualização dos resultados. A camada escondida ou intermediária são todas as camadas entre a de saída e a de entrada.

Diferentemente dos esquemas simbolistas normalmente utilizados nos sistemas de Inteligência Artificial convencionais, onde o conhecimento deve ser representado na forma de regras e/ou algoritmos, as Redes Neurais Artificiais aprendem através de exemplos, ou seja, o problema é modelado informalmente através da apresentação exaustiva de casos típicos.

A partir deste aprendizado, as redes tendem a generalizar o seu conhecimento e passam a responder corretamente a casos

[1] Segundo nomenclatura atualmente utilizada: *Redes Neurais* é a interconexão de elementos processadores que simulam o comportamento dos neurônios biológicos encontrados nas *Redes Neuronais*.

novos, desde que parecidos com os exemplos apresentados. Outras importantes propriedades das Redes Neurais advém do fato da "memória" da rede estar distribuida, ou seja, o conhecimento está espalhado pelos diversos neurônios na forma de pesos.

A distribuição de sua memória, torna as Redes Neurais Artificiais tolerante a falhas, ou seja, a perda de alguns neurônios, não trará como conseqüência a destruição de uma quantidade considerável de informação.

Tais propriedades, fazem com que as Redes Neurais Artificiais possuam um escopo muito grande de aplicações, dentre os quais podemos citar o setor elétrico [CIGRE93], [Souza *et. al.* 93]. Dentre as áreas consideradas promissoras para utilização de Redes Neurais no setor elétrico podemos destacar a diagnose de faltas e o processamento de alarmes.

O grande objetivo destas duas áreas é fornecer ao operador do Sistema Elétrico, informações precisas do estado atual do sistema, auxiliando-o a identificar e corrigir quaquer problema existente. Esta tarefa pode ser vista simplesmente como um problema de reconhecimento e classificação de padrões pois, ao reconhecer corretamente um padrão particular de alarmes, consegue-se caracterizar corretamente qual o problema que originou estes alarmes (figura 1).

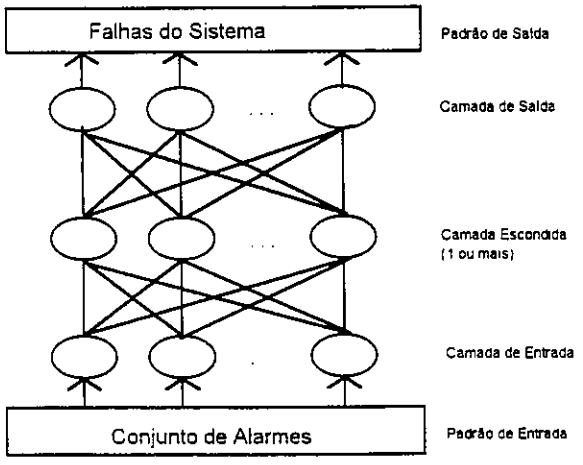


Figura 1 : Diagnose de Faltas como um problema de reconhecimento e classificação de padrões

A tarefa de reconhecer e classificar padrões é realizada facilmente com sucesso pelas Redes Neurais, inclusive devido a sua capacidade de distribuição de memória, ela consegue classificar padrões de entrada incompletos ou com ruído. É claro, que a precisão da classificação decai, mas ainda assim, é feita com sucesso.

Outra vantagem é a sua alta velocidade de processamento proporcionada pela capacidade de processamento paralelo (originada da densa rede de conexões entre os neurônios), requisito imprescindível para um rápido reconhecimento dos problemas existentes no sistema elétrico.

III - Aplicação em Diagnose de Sistemas de Potência:

Mostraremos como caso exemplo, um sistema elétrico simplificado de uma usina (figura 2). Este sistema elétrico é composto por duas unidades geradoras, dois transformadores e seus respectivos disjuntores, dois barramentos (barra principal e barra auxiliar) e dois alimentadores (também com seus respectivos disjuntores).

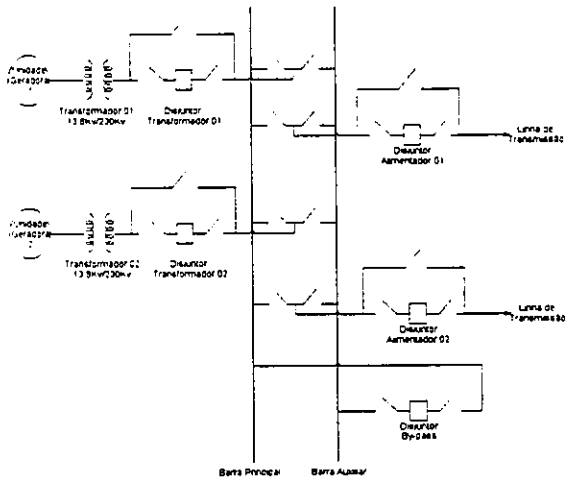


Figura 2 : Configuração do Sistema Elétrico de uma usina

Cada transformador possui três pontos de monitoração: Abertura do disjuntor (indica se o disjuntor está aberto ou não), Sobrecarga (mostra que ocorreu sobrecarga no

transformador) e Auxiliar (informa se o transformador está operando com o disjuntor de by-pass, ou seja, seu próprio disjuntor está em manutenção).

Já cada alimentador possui seis pontos de monitoração: Abertura do disjuntor (indica se o disjuntor está aberto ou não), Fase (mostra que ocorreu sobrecorrente na linha de fase), Terra (informa que ocorreu sobrecorrente na linha de terra), Temporizado (significa que ocorreu uma falha distante ao alimentador), Instantâneo (revela que ocorreu uma falha próxima ao alimentador) e Auxiliar (informa se o alimentador está operando com o disjuntor de by-pass, ou seja, seu próprio disjuntor está em manutenção).

A partir destes pontos de monitoração foi criada uma lista de descrição de problemas (falhas) deste sistema, onde cada falha está associado a um grupo de alarmes. Deste modo, a falha próxima ao alimentador 01 f/f é caracterizada pela atuação dos alarmes: abertura do disjuntor do alimentador 01, fase do alimentador 01 e instantâneo do alimentador 01 e assim por diante.

Durante a implementação, nosso objetivo será utilizar este exemplo para procurar uma explicação para um conjunto de alarmes atuados. Assim, quando o sistema apresentar os seguintes alarmes atuados: abertura do disjuntor do alimentador 01, terra do alimentador 01 e instantâneo do alimentador 01, ele deve ser capaz de diagnosticar que uma falha próxima ao alimentador 01 f/t está ocorrendo.

Devemos destacar, que o sistema deve ser capaz de diagnosticar não somente falhas simples mas também falhas múltiplas. Além, de ser apto a diagnosticar falhas mesmo quando o conjunto de alarmes que a caracterizam estiver incompleto. Deste modo, se somente os alarmes terra do alimentador 02 e temporizado do alimentador 02 estiverem atuados, o sistema deve ter a habilidade de diagnosticar que a falha distante ao alimentador 02 f/t está ocorrendo mesmo sem a presença do alarme de abertura do disjuntor do alimentador 02 já que esta falha é a que melhor justifica a atuação conjunta destes dois alarmes. Devemos notar, que esta característica aumenta consideravelmente a complexidade do problema.

III.1 - Representação do Sistema:

Baseado no sistema elétrico mostrado acima, implementamos um processador inteligente de alarmes empregando o modelo "back-propagation", onde cada neurônio de entrada representa um alarme e cada neurônio de saída representa uma falha do sistema.

A forma de representação escolhida para os vetores de entrada e saída foi a binária. Assim, quando queremos representar que um alarme está ativo ou não, basta associá-lo aos valores 1 e 0 respectivamente.

Para ensinar a rede, foi criado um arquivo de treinamento, que continha falhas simples e duplas num total de 238 padrões (deve-se notar que denominamos de falhas duplas, a conjunção de duas falhas simples). Este arquivo foi então, apresentado a rede neural (figura 3) e seu treinamento foi realizado utilizando o software de simulação de rede neural NeuralWorks Professional II/Plus [Neuralware 91] em um micro-computador IBM PC/AT compatível.

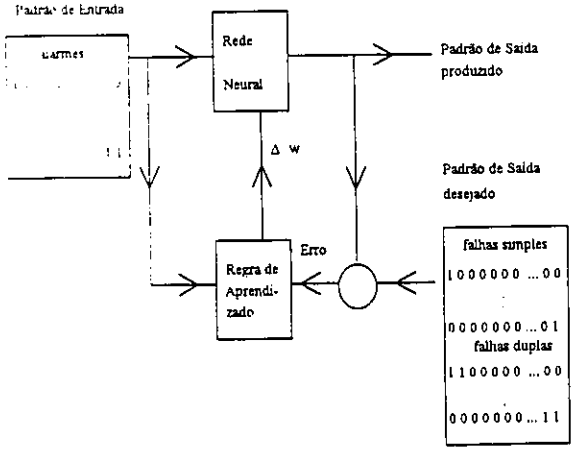


Figura 3 : Treinamento de um processador inteligente de alarmes para o sistema elétrico simplificado de uma usina

III. 2 - Performance:

A partir do arquivo de treinamento, passamos a testar neste software, quais os

parâmetros que poderiam dar uma melhor performance para a rede.

Primeiramente, procuramos descobrir o melhor dimensionamento da rede. Para isto, variamos apenas o número de camadas intermediárias (uma ou duas) e o número de neurônios em cada uma dessas camadas (5, 10, 20 e 30 neurônios).

Os resultados demonstraram que o erro de aprendizado utilizando duas camadas intermediárias era 10% maior que o empregando apenas uma camada e neste grupo, a melhor performace foi obtida pelo treinamento com 5 neurônios na camada intermediária (figura 4).

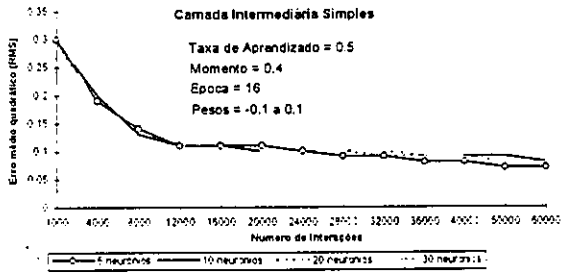


Figura 4 : Resultados de aprendizado com uma única camada intermediária

Passamos então, a variar outros parâmetros caracteristicos da rede (taxa de aprendizado, momento e época) com o objetivo de diminuir o erro de aprendizado aprimorando ainda mais o resultado do treinamento.

Inicialmente, foram testadas várias taxas de aprendizado e a melhor performance foi obtida pela taxa de aprendizado igual a 0,90 (figura 5).

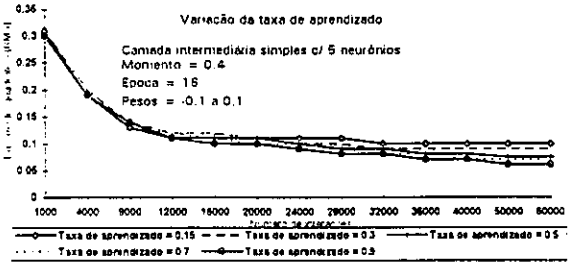


Figura 5 : Resultados variando a taxa de aprendizado

Em seguida, variamos o valor do momento e o momento com valor igual a 0,8 obteve o melhor resultado, fazendo cair significativamente o erro (figura 6).

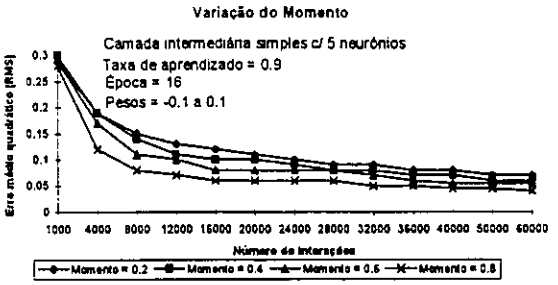


Figura 6 : Resultados variando o momento

Finalmente, passamos a alterar o valor da época(tamanho do conjunto de padrões de entrada após o qual deve ser atualizado os pesos das ligações) e o menor erro rms de aprendizado foi obtido pela época igual a 11 (figura 7).

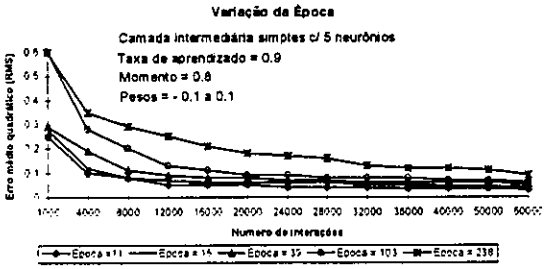


Figura 7: Resultados variando o valor da época

Deste modo, os parâmetros da rede que obtiveram o melhor resultado no treinamento foram : Camada intermediária simples com 5 neurônios, Taxa de aprendizado = 0,9 , Momento = 0,8 , Época = 11 e Intervalo dos pesos entre -0,1 a 0,1.

Estes resultados foram obtidos durante o treinamento da rede mas para verificar se a rede realmente aprendeu, precisamos testá-la não só com os padrões de treinamento como também, com outros padrões diferentes denominados de padrões de teste para verificar a capacidade de generalização da rede.

Os testes foram realizados com quatro tipos de padrões diferentes :

- treinamento → contém todos os padrões de treinamento (238 padrões).

- falhas múltiplas → contém todas as falhas triplas e quádruplas possíveis, onde classificamos de falhas triplas e quádruplas, a conjunção de três e quatro falhas simples diferentes (580 padrões).
- falhas simples com ruído → contém todas as falhas simples com ruído na entrada. Deve-se notar que denominamos de ruído, a ausência de um dos alarmes que caracterizasse as respectivas falhas (92 padrões).
- falhas duplas com ruído → contém falhas duplas com ruído na entrada (745 padrões).

Durante estes testes, a rede apresentou uma taxa de erro de 2,96% sendo este resultado bastante promissor, mostrando que a rede é capaz de diagnosticar falhas mesmo quando submetida a novos padrões (padrões diferentes dos padrões de treinamento).

Devemos também destacar que o tempo de resposta da rede neural independe do número de alarmes ativados, sendo este de aproximadamente um segundo.

IV - Conclusão:

Os resultados obtidos foram bastante satisfatórios pois quando não ocorre ausência de alarmes, a taxa de acerto é de 100 % entretanto quando o conjunto de alarmes estava incompleto a taxa global de acerto decaiu para 97%, o que é ainda bastante bom.

Isto nos encoraja a dar prosseguimento a este trabalho. Para tal, novos tópicos devem ser avaliados para melhorar o diagnóstico como a caracterização da ordem temporal dos alarmes bem como a implementação de um sistema híbrido (Conexionista + Simbolista) onde a rede neural informa quais os problemas que estão ocorrendo e um sistema especialista explica os motivos associados a cada problema. Inclusive, um sistema especialista utilizando lógica não-monotônica já foi desenvolvido com sucesso [Silva 94] para este mesmo caso exemplo faltando apenas realizar sua integração com a rede neural.

V - Bibliografia :

[CIGRE 93] CIGRE, "Artificial Neural Networks for Power Systems - A Literature Survey", CIGRE - 1993 Session, July 1993, Ref. Task Force 38.06.06.

[NeuralWare 91] NeuralWare Inc., "NeuralWorks Professional II/Plus and Neural Works Explorer - Reference Guide", 1991.

[Kim *et. al.* 91] K. Kim, J. Park, "Application of Hierarchical Neural Network to Fault Diagnosis of Power Systems", Third Symposium on Expert Systems Applications to Power Systems, Tokyo, April 1991, pp 323-327.

[Rodriguez *et. al.* 92] C. Rodriguez, S. Rementeria, C. Ruiz, A. Lafuente, J. I. Martin, J. Murgueza, "A Modular Approach to the Design of Neural Networks for Fault Diagnosis in Power Systems", International Joint Conference on Neural Networks, Baltimore, June 1992, vol.III, pp 16-23.

[Silva 94] Victor N. A. L. da Silva, "Diagnose em Sistemas de Potência utilizando Lógica Não-monotônica e Redes Neurais"), Tese de M. Sc., COPPE-UFRJ, Abril 1994.

[Souza *et. al.* 93] G. N. F. de Souza, N. de M. Roehl, M. Moszkowicz, "Experiência de Aplicação de Redes Neuronais ao Setor Elétrico", V ERLAC - Encontro Regional Latino-Americano da CIGRE, 1993.

[Tanaka *et. al.* 89] H. Tanaka, S. Matsuda, H. Ogi, Y. Izui, H. Taoka, T. Sakaguchi, "Design and Evaluation of Neural Network for Fault Diagnosis", Second Symposium on Expert System Applications to Power Systems, Seattle, Washington, July 1989, pp.378-384.

[Wollenberg 86] B. F. Wollenberg, "Feasibility for an Energy Management System - Intelligent Alarm Processor", IEEE Transactions on Power Systems, Vol. 1, No. 2, May 1986, pp. 241-247.

1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

UM SISTEMA HÍBRIDO PARA A PREVISÃO DE CARGA A CURTO-PRAZO

Germano Lambert Torres[1]
Alexandre Pinto Alves da Silva[1]
Jamil Haddad[1]
Luiz Octávio Mattos dos Reis[2]

1 - ESCOLA FEDERAL DE ENGENHARIA DE ITAJUBÁ
2- UNIVERSIDADE DE TAUBATÉ

Abstract - **Este artigo apresenta um método alternativo para a previsão da carga elétrica por barramento ou por área de um sistema elétrico de potência. Este método é composto por um sistema híbrido que comporta uma rede neural treinada para este tipo de ãtividade e um sistema especialista difuso para um melhor ajuste da previsão. O artigo apresenta uma revisão dos métodos clássicos de previsão e exemplos do método proposto com dados reais de um sistema elétrico.**

Keywords - **Previsão de carga, operação de sistemas elétricos, redes neurais, sistemas especialistas difusos.**

I. INTRODUÇÃO

Durante o planejamento de um sistema elétrico é necessário prever o crescimento da carga para se poder construir um sistema de geração-transmissão que seja capaz de fornecer toda a demanda requerida. A grande dificuldade neste tipo de previsão é o tempo envolvido pois deve ser realizado em planos decenais ou quinquenais, podendo ser a previsão alterada por diversos fatores, principalmente, os econômicos-financeiros, sociais e ecológicos.

Durante a operação dos sistemas elétricos de potência deve-se harmonizar a geração com o consumo de energia elétrica, ou seja, deve-se somente gerar o que está sendo efetivamente consumido, pois um desbalanço nesta equação poderia causar problemas de elevação ou queda de tensão no sistema. Com isto, o operador do sistema deverá ser capaz de saber a carga que será consumida antecipadamente para acionar o sistema de geração. Atualmente, os operadores dispõe de alguns recursos empíricos para realizar tal previsão.

Todavia, com ao aumento e a diversificação da carga e com a aponsentadoria de vários operadores experientes, está cada vez mais difícil de realizar tal previsão com uma margem pequena de erro.

Existem dois diferentes objetivos para o estudo da previsão da carga elétrica são eles: a operação econômica das usinas de geração de energia e uma solução tecno-econômica dos sistemas de transmissão e distribuição. O primeiro objetivo visa melhorar o desempenho global do sistema sem levar em consideração cada barra de carga. O segundo objetivo trabalha com as características de um dado barramento ou área do sistema.

Pode-se classificar as diversas previsões de carga baseadas no fator tempo. Esta classificação é feita da seguinte maneira:

a) curto-prazo - dos próximos quinze minutos até uma semana a frente,
b) médio-prazo - de uma semana a um ano a frente.
c) longo-prazo - após um ano.

Os tempos discriminados acima são orientativos podendo segundo a conveniência ou o tipo de

'departamentalização da concessionária por ser alterado.

Os critérios que são importantes para a previsão em cada um dos intervalos acima variam. Por exemplo, para uma previsão a curto-prazo podem ser importantes os seguintes fatores: temperatura, hora do dia, estação do ano, humidade, entre outros. Enquanto, para uma previsão de longo prazo aspectos econômicos e políticos devem ser levados em consideração.

Também os objetivos para cada tipo de previsão são diferentes. Em uma previsão de curto-prazo deseja-se compatibilizar a geração com a carga, enqaunto na de médio-prazo, o objetivo pode ser realizar um programa de manutenção do sistema ou de otimizar as fontes de energia primária utilizadas.

Este artigo apresenta inicialmente uma revisão dos métodos clássicos de previsão da carga a curto-prazo. Em seguida, o sistema proposto é descrito, sendo feita uma análise de cada um de seus componentes. Depois, é apresentado a interação deste sistema em um centro de operação e seu relacionamento com os operadores. Finalmente, alguns exemplos utilizando dados reais são apresentados e o desempenho do sistema é analisado.

II. REVISÃO DOS MÉTODOS DE PREVISÃO DA CARGA ELÉTRICA A CURTO-PRAZO

A adaptação da geração a carga deve levar em consideração as caracteristicas de cada uma e, também, dos caminhos que podem fazer esta ligação, ou seja, o sistema de transmissão.

As curvas de carga podem ser representadas por valores horários, diários ou por ciclos semanais. Os valores destas curvas podem ser afetadas por fatores climáticos ou sazonais ou ainda por fatores cíclicos semanais, mensais ou anuais.

Por outro lado, as características da geração podem ser afetadas por elementos previsíveis, tais como períodos de seca ou manutenção preventiva, e por elementos aleatórios, como uma saída forçada da geração.

Os caminhos entre a geração e a carga podem ser achados através de processos lógicos ou buscas heurísticas. Esta seção apresenta a formulação clássica do problema e as técnicas de solução.

II.1 Formulação do Problema

A previsão de carga pode ser dividida em dois tipos: modelagem da carga máxima e modelagem da curva de carga [1]. Enquanto o primeiro expressa somente o valor máximo diário, semanal ou mensal, o segundo modela todo o comportamento da curva. Neste artigo apresenta-se um procedimento alternativo para o segunda tipo de modelagem.

Na modelagem da curva de carga, o valor previsto F(t) é obtido através da operação entre a curva padrão de carga, B(t), e de uma curva de desvio, W(t). Esta operação pode ser aditiva ou multiplicativa segundo a obtenção da curva de desvio.

$$F(t) = B(t) + W(t) \tag{1}$$

$$F(t) = B(t) \bullet W(t) \tag{2}$$

a. Curva Padrão da Carga - Este valor deve caracterizar a carga de base da barra, área ou sistema a ser previsto. É calculada utilizando dados históricos, que são agrupados segundo critérios hora-do-dia, dia-da-semana e tempo-no-ano (semanal-mente, mensalmente ou sazonalmente). Esta divisão permite separar cargas que tenham o mesmo comportamento, por exemplo, separar sábados e domingos como dias da semana, se for o caso.

A carga padrão pode ser dividida em duas partes. A primeira utilizando a média da carga dos dias comuns do período. No caso estudado, por exemplo, os feriados tinham comportamento semelhante aos sábados e domingos e eram, portanto, reunidos com estes dias. A segunda parte expressa características particulares de cada dia da semana individualmente. Isto pode ser feito através de uma média móvel simples ou ponderada. As equações (3) a (5) apresentam o cálculo da carga padrão.

$$B(t) = B_1(t) + B_2(t) \tag{3}$$

$$B_1(t) = \frac{1}{m.n} \sum_{w=1}^{m} \sum_{d=1}^{n} L(t,d,w) \tag{4}$$

$$B_2(t) = \frac{1}{n}\sum_{w=1}^{m} L(t,d,w).B_1(t) \qquad (5)$$

Onde L(t.d.w) representa a carga em um tempo t para o dia-da-semana d da semana w. O valor de n pode mudar de uma semana para outra e representa a homogeinidade dos dias da semana. O valor de m representa o número de semanas do período que está sendo modelado.

Outros fatores podem ser somados a equação (3) dependendo da correlação existente entre os dados utilizados. Por exemplo, em [2], propõe-se a inclusão de uma parcela de polinômios de segunda ordem e componentes dependentes da temperatura.

b. Curva de Desvio da Carga - Este valor é utilizado para representar as variações do desempenho recente da carga. Este valor contém informações sobre as últimas horas ou até últimos dias. Alguns métodos utilizados para calcular este tipo de carga são a autoregressão ou o alisamento exponencial.

II.2 Revisão dos Trabalhos

Gross e Galiana, em [3], dividiram a previsão da curva de carga em horária e baseada em modelos dinâmicos. No primeiro, a curva de carga é obtida baseada nos dias da semana e nas condições climáticas, bastando ao operador selecionar qual a curva que mais se adapta ao dia em questão. No segundo tipo, a previsão da carga não é somente função da hora do dia mas também das informações exógenas ao sistema.

Lambert Torres *et al.*, em [4], propuseram um sistema especialista para calcular a carga prevista Neste encaminhamento, técnicas de matematica formal são agregadas à experiência do operador, resultando em regras representadas por proposições condicionais difusas.

Monghram e Rahman, em [5], avaliaram cinco diferentes métodos de previsão: regressão multilinear, séries estocásticas (média móvel integrada autoregres-siva), alisamento exponencial, filtros de Kalman e sistemas especialistas. Cada método foi aplicado a mesma base de dados, permitindo uma compração direta dos resultados.

III. APRESENTAÇÃO DO MÉTODO PROPOSTO

Como foi visto na seção anterior, existem várias maneiras para se prever a carga elétrica, todavia, em todos eles, é desejável a montagem de uma base de dados que reflete o histórico da carga, bem como aspectos climáticos relevantes. Esta seção apresenta inicialmente a estrutura da base de dados construída, para, em seguida, mostrar o sistema proposto.

III.1 Estrutura da Base de Dados

A base de dados que contém informações históricas sobre o comportamento da carga e os parâmetros meteriológicos foi montado. Dados, tais como : temperatura a bulbo seco e molhado, humidade relativa e direção e intensidade do vento, foram alguns dos elementos que compunham esta base. Além desses, informações especiais como inclinação do sol e nebulosidade também podem afetar a carga, principalmente, a residencial.

A estrutura da base de dados foi a seguinte, onde v representa o valor da grandeza, h, a hora, d, o dia, w, a semana do ano, y ,o ano, e s a estação:

Potência Ativa :	$P(v,h,d,w,y,s)$
Temperatura a bulbo seco:	$\Theta(v,h,d,w,y,s)$
Temperatura a bulbo molhado:	$M(v,h,d,w,y,s)$
Humidade Relativa :	$H(v,h,d,w,y,s)$
Direção do vento:	$WD(v,h,d,w,y,s)$
Velocidade do vento:	$WS(v,h,d,w,y,s)$

Os valores de potência real, valores de temperatura, humidade relativa e velocidade do vento são armazenados em [MW], [°C], [%] e [km/h], respectivamente. Para os valores de direção do vento, emprega-se a convenção a seguir: Norte é 1, e cada 22,5° no sentido horário é 1 mais. Para dia calmo usa-se 0. Assim, por exemplo, a direção WSW é 12 e NE é 3.

Utiliza-se a notação 24 horas para valores horários. Os valores de dia são numerados de 1 para segunda-feira a 7 para Domingo. Os feriados recebem o número 6 (Sábado). Os valores de semana são numerados por estação e o mais recente recebe o número 1. Este banco de dados é construido para manipular usando os últimos três anos e o ano atual.

Os valores de ano são de 1 para o ano atual a 4 para três anos depois. Os valores de estação são os seguintes: inverno - 1, primavera - 2, verão - 3, e outono - 4.

Para até dois valores em falta (v) dos mesmos dados, eles são calculados usando-se média simples entre o valor anterior e o próximo valor. Quando há mais de dois valores faltando, os dados são esquecidos.

Dias especiais (como tempestade de neve) são armazenados no banco de dados com um "flag". Estes valores não são usados para os cálculos de corrente, exceto quando esta condição de tempo é prevista.

III.2 Descrição do Método

O método proposto faz a previsão de carga de curto-prazo utilizando um modelo híbrido com técnicas de redes neurais e sistemas especialistas. Inicialmente, a rede tem o objetivo de realizar a melhor previsão possível enquanto o sistema especialista vai adequar este valor previsto a fatos ou características especiais da carga. A figura 1 apresenta o sistema híbrido proposto.

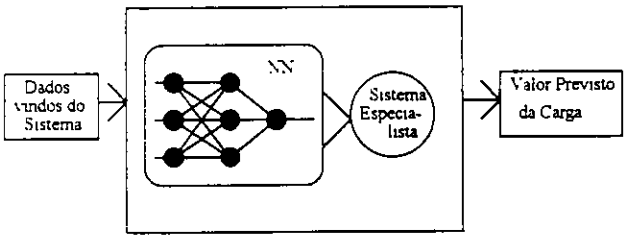


Figura 1 - Sistema Híbrido Proposto

A estrutura usada é um modelo de camada com três camadas (camada "input", camada "hidden" e camada "output"). A camada "input" é composta de um número de neurons igual ao número de dados de informação sobre a carga e seus parâmetros de tempo. A temperatura de bulbo seco, a temperatura de bulbo úmido, a umidade relativa, a direção do vento, a velocidade do vento, são os parâmetros de tempo mais comuns. Para cargas especiais, outros fatores (como inclinação do s... nebulosidade) também podem exercer influência sobre a carga.O número de insumos depende do tipo de barramento, tipo de previsão de carga (uma hora a frente, duas horas a frente) e a hora do dia. A camada "output" é composta de um ou mais neurons. Estes neurons dão os valores de previsão de carga. Através de uma regra empírica, o número de neurons da camada "hidden" é uma média entre as camadas "input" e "output".

No método proposto, os pesos são calculados por um processo de aprendizado usando-se a retro-propagação do erro. O processo de aprendizado é feito usando-se dados medidos de barramento. Um conjunto inicial de dados da "input" é apresentado ao sistema neural que ajusta os valores de peso para um erro mínimo. A seguir, é apresentado um novo conjunto de dados iniciais e os valores de peso são ajustados consoantemente. O processo acaba quando a diferença entre "output" alvo e "output" encontrado para todos os conjuntos "input" está próxima de zero.

O sistema especialista visa ajudar na previsão da carga em duas situações comportamento não convencional da carga devido a fatores endógenos e exógenos. Os fatores endógenos são quando a situações pouco comum, normalmente, das condições meteriológicas, como por exemplo, tempestades de neve ou valores de temperatura e/ou humidade atípicos.

Os fatores exógenos refletem hábitos e comportamentos dos consumidores face a variadas situações, como por exemplo a rotina diária e a inércia térmica.

A carga de rotina diária pode ser descrita como atitudes usuais para uma parte específica do dia. Por exemplo, se houver um ponto de carga a 9:00 am e a carga as 8:00 am for maior do que a carga padrão para as mesmas condições de tempo, então parte do ponto de carga já foi consumida. Da mesma forma, se a carga as 8:00 am for menor que a carga padrão então, possivelmente o ponto de carga será aumentado. Esta forma de carga residual é aplicável em previsão de carga para até 3 horas a frente da previsão

A inércia térmica deverá ter dificuldade na mudança de uma temperatura de melhor para pior. Por exemplo, se a temperatura média no inverno para o último dia era -5°C, e hoje é de -20°, a carga aumentará devido a diferença de temperatura mais uma certa inércia térmica. Se pelo próximo dia, a temperatura permanece em -20°C, este componente inercial diminuirá ou transformará-se em nada. No Canadá, por exemplo, este fator é muito importante devido a faixas de temperatura excessivas. Esta forma

de carga residual é principalmente aplicável para previsão de carga 24 horas a frente.

IV. EXEMPLO ILUSTRATIVO

Este exemplo utiliza dados da Hydro-Quebec Power System, Canadá. Os dados numéricos vão de Abril/1984 a Julho/1988 e contêm informações relacionadas aos fluxos de potência real nas linhas 1288 e 1289, bem como condições de tempo tais como temperatura de bulbo seco, temperatura de bulbo úmido, humidade relativa, velocidade e direção do vento. Estes parâmetros foram medidos por hora. O desempenho mostrado neste item é função da adição destas duas linhas porque elas têm funcionado em paralelo. A estação usada neste exemplo foi o inverno, e o o objetivo era computar a carga prevista em 20 de Janeiro de 1988, às 10 e 11 horas.

Para isto, foi usada a estrutura proposta no ítem anterior. Esta estrutura permitiu treinar uma rede neural para muitos "inputs". Os "inputs" e padrões de treinamento devem ter valores numéricos entre 0 e 1, logo pode-se fazer uma normalização destes conjuntos usando-se valores mínimos e máximos. O Apêndice apresenta algumas redes neurais treinadas para diferentes conjuntos de "input". São apresentados os "inputs" usados, os pesos e a ativação interna para cada neurônio. Por exemplo, no exemplo 1 (tabela 1) os "inputs" são:

P(9) - potência real as 9 horas
Θ(9) - temperatura de bulbo seco as 9 horas
H(9) - umidade relativa as 9 horas
Θ(10) - temperatura de bulbo seco as 10 horas
H(10) - umidade relativa as 10 horas

O número de neurônios na "hidden" foi o mesmo do "input" usado. Após o treinamento das redes neurais podemos encontrar os valores de previsão de carga usando os valores de 20 de janeiro como dados de "input". Os valores alvo são 165[MW] e 170[MW] as 10 e 11 horas, respectivamente. A tabela abaixo apresenta estes valores para cada exemplo.

V. CONCLUSÃO

Foi desenvolvida uma abordagem alternativa para computação do valor de previsão de carga usando-se técnicas de rede neural. Esta abordagem utiliza a técnica da retropropagação do erro para fazer o aprendizado da rede. A carga prevista obtida é tratada por um sistema especialista e o resultado é superior aos métodos correntes em sistemas de potência. Outra vantagem deste sistema é sua velocidade em encontrar o valor previsto.

Tabela - Resultado de Diversas Redes

Exemplo	10 horas	Erro (%)	11 horas	Erro (%)
1	165.6	0.36	-	-
2	169.1	2.48	-	-
3	158.8	-3.76	-	-
4	164.3	-0.42	-	-
5	165.1	0.09	166.5	-2.06
6	162.4	-1.57	167.0	-1.75

VI. AGRADECIMENTOS

Os autores agradecem o Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo suporte financeiro no desenvolvimento deste trabalho.

VII. REFERÊNCIAS

[1] G. Lambert-Torres, B. Valiquette & D. Mukhedkar - "Bus Load Forecasting using Fuzzy Techniques", CEA Enginering and Operating Transactions, Vol. 29, Part IV, 1990.

[2] M.S. Abu-Hussien *et al.* - "An Accurate Model for Short-term Load Forecasting", IEEE Trans. on Power Apparatus System, Vol. PAS-100, No.9, 1981, pp.4158-4165.

[3] G. Gross & F.D. Galiana - "Short-term Load Forecasting", Proceedings of IEEE, Vol. 75, No. 12, 1987, pp. 1558-1573.

[4] G. Lambert-Torres, B. Valiquette & D. Mukhedkar - "Short-term Feeder Load Forecasting: An Expert System using Fuzzy Logic", International Symposium on Power Systems and Power Plant Control, IFAC, Korea, Aug. 22-25, 1989, pp. 317-322.

[5] I. Moghran & S. Rahman - "Analysis and Evaluation of Five Short-Term Load Forecasting Techniques", IEEE Winter Meet., 89WM171-0 PWRS, 1989.

APÊNDICE

Exemplo 1

Inputs: P(9), θ(9), H(9), θ(10), H(10)
Output: P(10)

	Weight					Threshold
h1	-5.978927	-1.653212	-2.548971	0.332980	-0.147354	2.936591
h2	-0.114415	-0.654037	0.335414	-0.760783	-0.110014	0.067189
h3	-1.005803	0.576769	1.103390	1.669233	0.716206	0.981162
h4	-1.784045	-1.827001	-1.579652	-2.149607	-1.492007	-0.693183
h5	-3.573369	1.045426	1.434333	0.637234	-0.653510	-1.586497
o1	-6.679025	-0.988277	-1.749346	3.127510	3.903967	-0.607988

Exemplo 2

Inputs: P(9), θ(9), M(9), H(9), θ(10), M(10), H(10)
Output: P(10)

	Weight						Threshold
h1	0.876312	-1.063275	-1.686141	-0.336451	-1.038807	-1.096977	-0.211819
	-0.853785						
h2	1.220775	0.336993	-0.971611	0.618897	0.767455	-0.528810	0.340267
	1.007709						
h3	0.820547	-0.835026	-0.853673	-0.435869	-1.129255	-0.889814	-0.436630
	-0.885352						
h4	-0.942197	-1.024489	1.156638	-0.298165	-0.396664	0.088166	-0.143552
	-0.452388						
h5	-1.076023	-0.372056	0.558461	-0.753959	-0.770369	0.352035	-0.149251
	-0.121903						
h6	1.062982	-0.952319	-1.567257	-0.842092	-0.576939	-1.469840	-0.382589
	-0.497015						
h7	5.240825	-1.401142	2.668851	-2.580875	-0.204704	1.741040	0.843365
	-2.337672						
o1	2.051456	1.705301	1.450081	-1.747807	-1.422895	2.055566	-0.267822
	6.348610						

Exemplo 3

Inputs: P(9), θ(9), M(9), θ(10), M(10)
Output: P(10)

	Weight					Threshold
h1	-7.003637	-1.773186	0.610258	-0.307218	-1.059579	2.398821
h2	-0.522576	-1.149151	0.585113	-0.876102	-0.715830	1.161999
h3	-2.485600	-1.701570	0.812569	-0.267826	-0.682070	0.778366
h4	2.325455	-1.795726	-2.547008	-2.211357	-2.672765	-0.301740
h5	4.123282	0.794142	-0.056664	0.801774	0.804435	-1.505060
o1	-6.547663	-2.537905	-3.336793	4.175916	3.485125	-1.020323

Exemplo 4

Inputs: P(9), θ(9), θ(10)
Output: P(10)

	Weight			Threshold
h1	-6.120185	-1.318981	-0.906765	1.556140
h2	-1.370813	3.436532	3.497077	1.550174
h3	8.038928	0.770676	1.238047	-1.853936
o1	-5.505402	-5.062915	6.317791	-0.159444

Exemplo 5

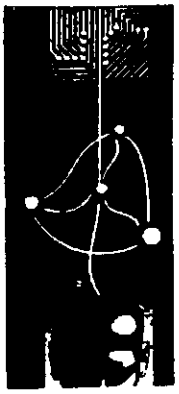
Inputs: P(9), θ(9), H(9), θ(10), H(10), θ(11), H(11)
Output: P(10), P(11)

	Weight						Threshold
h1	-2.850067	0.064379	-1.867121	0.949502	-3.003542	1.033283	1.440434
	-0.010479						
h2	-0.290933	-0.813180	-0.616699	-0.552076	0.201466	-0.221277	0.393604
	1.467901						
h3	1.286221	-1.622224	-0.696168	-0.988542	-0.837983	-1.605588	-0.534564
	-1.028042						
h4	1.120417	-0.014424	1.053302	0.506672	1.160000	-0.451046	-0.597388
	-0.410352						
h5	0.700442	-0.409511	0.161902	-1.180542	-0.017419	-0.446406	-0.179351
	0.291033						
h6	3.139088	0.800937	1.169884	0.748396	2.949975	-1.142437	-1.470714
	-2.465890						
h7	-1.853428	1.738360	0.830380	1.488542	1.120508	2.205424	0.604597
	0.560077						
o1	-3.036061	-1.803710	2.203408	1.419467	-0.528269	4.697513	-0.069237
	-3.548041						
o2	-3.836864	0.420621	1.148244	1.296347	1.080802	1.605404	0.003898
	-2.559558						

Exemplo 6

Inputs: P(8), θ(8), H(8), P(9), θ(9), H(9), θ(10), H(10), θ(11), H(11)
Output: P(10), P(11)

	Weight					Threshold
h1	-1.792827	0.831928	1.794066	-1.088477	-0.405940	-0.754631	-0.131692
	-0.586769	-1.530238	0.339374	-0.065960		
h2	-0.934255	-0.026349	1.135989	-0.421337	0.297041	-0.083328	-0.168162
	-0.635903	-1.086967	-0.193668	0.486377		
h3	0.177775	0.009621	0.112965	-0.052116	-0.941504	0.224031	-0.228416
	-0.925864	0.843280	-0.585827	1.758739		
h4	0.695570	-0.136292	-1.092583	0.405205	-0.627116	0.523179	-0.007909
	-0.429530	1.270333	0.213709	0.319345		
h5	-0.729748	-0.420629	0.937431	-0.126993	-0.572058	-0.280580	-0.223458
	-0.433243	-0.053117	0.341390	0.771380		
h6	0.303579	-0.031947	-0.430923	0.471736	0.422501	0.204312	0.058901
	-0.269793	-0.038239	.0104715	0.113485		
h7	-1.474764	0.527824	1.160007	-1.356912	-0.035175	0.307257	-0.058143
	-0.020537	-1.425624	0.310711	-0.561995		
h8	1.101532	-0.664641	-1.535768	1.276018	0.379529	0.930027	0.142649
	0.247574	1.315829	-0.062923	0.067539		
h9	1.038968	-1.428416	-0.923639	0.619814	-0.794065	-1.171629	-0.422484
	-0.251820	-0.579720	-0.980323	-0.219754		
h10	-0.093055	-1.002639	-0.747140	0.626757	-0.626616	-0.097610	0.132451
	-0.248395	-0.122497	-0.281901	-0.509798		
o1	-3.028203	-1.786724	-0.991701	0.808280	-1.532955	0.344811	-0.356970
	-2.173898	2.457000	1.927562	0.410490		
o2	-2.123776	-0.887667	1.722687	1.319648	-0.129423	-0.093988	-1.517802
	-2.434891	0.745597	1.541783	0.742043		



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajuba
Itajuba, 24 a 27 de outubro de 1994

MONITORAÇÃO EM TEMPO REAL DE POTÊNCIA ELÉTRICA DE INTECÂMBIO USANDO REDE NEURONAL ARTIFICIAL

Pedro Rodrigues de Brito Filho
Jurandyr Nascimento Garcez
Depto de Engenharia Eletrica - Centro Tecnologico
Universidade Federal do Para
Caixa Postal - 8619 Fax - (091) - 229-7491

Wady Charone Júnior
Centrais Eletricas do Norte do Brasil - S.A
ELETRONORTE
Av. Perimetral s/n - Terra Firme
Fax - (091) - 224-6720

RESUMO

Este trabalho apresenta uma aplicação de Rede Neuronal Artificial (RNA) como um suporte na operação do Sistema Eletrico Norte-Nordeste (SE N/Ne) da Eletronorte direcionada a supervisão da potência eletrica de intercâmbio entre as areas Eletronorte e CHESF. A RNA multicamadas treinada pelo metodo do Back-Error-Propagation permite tomadas de decisões rapidas e confiaveis quanto ao nivel de geração em Tucurui, ja que a resposta da RNA e praticamente instantânea O software foi desenvolvido e testado na Rede Local de Supervisão (RLS) do CORB da Eletronorte em Belem apresentando resultados promissores

Palavras Chave: Sistema de Potência - Supervisão em Tempo Real - Inteligência Artificial - Redes Neuronais Artificiais - Potencia de Intercambio

1. INTRODUÇÃO

Atualmente a Inteligência Artificial, tem sido uma alternativa com grande possibilidade de sucesso na soluções de problemas operacionais em sistemas eletricos de potência Os Centros de Controle de Operação (COS's) estão seguindo este caminho vizando a modernização para atender o aumento substancial nas demandas e nas interligações. Atraves de cooperação tecnica entre a Eletronorte e a UFPA estão sendo desenvolvidos estudos para implementação de sistemas inteligentes baseados em Redes Neuronais Artificiais (RNA) na tentativa de se obter meios de automatização do COS. Neste trabalho mostra-se mais um passo dado neste rumo a monitoração da potência eletrica de intercâmbio. Para sua realização, adotou-se um fluxograma de regulamentação para não ferir o programa adotado pela empresa quanto a automatização, tanto a nivel de "hardware" como de "software". Portanto, o processo de obtenção dos dados em tempo real da-se de forma a obedecer a todo um procedimento que rege o sistema de base, no caso, uma Rede Local de Computadores, conhecida aqui por Rede Local de Supervisão (RLS)

Aquisição, tratamento, memorização na base de dados visualização de resultados e imagens constituem funções basicas do Sistema de Supervisão e Controle Atraves desta cadeia, esta disponivel o estado do sistema eletrico em tempo real

De posse deste estado pode-se implementar diversas funções tais como: análise de redes eletricas em tempo real; sistemas especialistas para tratamento de alarmes; geração e tratamento de alarmes usando RNA; relatorios estatisticos e operacionais diversos; interface homem-maquina amigavél; perturbografia; logger de eventos, etc [1]

Um dos problemas serio que ocorrem em sistemas de potência e a ultrapassagem do limite de estabilidade, que pode provocar grandes danos caso não seja prontamente solucionado. Para se aumentar a confiabilidade do sistema deve-se tomar providencias no sentido de que sua operação não seja muito proxima deste limite. Em dada epoca do ano, o SE N/Ne opera proximo ao seu limite de estabilidade Para evitar a operação em um nivel de severidade mais critico, os operadores devem processar uma significativa quantidade de informações afim de que o sistema não entre em colapso A aplicação de uma RNA gerou uma ferramenta para auxiliar os operadores quando da necessidade de tomada de decisões, tratando com mais rapidez e eficiência as informacoes primordiais para operação do sistema

2. O AMBIENTE COMPUTACIONAL

O Sistema de Supervisão e Controle (SSC) da Eletronorte está passando por um processo de modernização onde atualmente todas as funções inerentes ao SSC estão centralizadas em um computador BULL-SEMS SOLAR 16-65 com outro idêntico servindo de "HotBackup". Lançando mão da ideia do processamento paralelo. estão sendo desenvolvidos em uma RLS todas as funções de supervisão do SSC. facilitando com isso o acesso ao "software" e "hardware".

A configuração inicial da RLS é fisicamente uma rede local de topologia em barramento (Tokem Bus) padrão ARCNET. a 2.5 megabits.

Capacidade multitarefa plena com priorização de tarefas. comunicação eficiente entre tarefas e respostas em tempo real em alta velocidade. são requisitos indispensáveis para se obter bons resultados das aplicações como as encontradas em processos industriais que impõem demandas pesadas de informações. Devido o fato de que o sistema operacional QNX da "QNX Software System LTD" de utilização crescente em controle de processos industriais na América do Norte e na Europa [2]. oferecer essas possibilidades. optou-se por utiliza-la na RLS.

A interligação física do SSC com a RLS. a distribuição inicial da RLS no sistema Eletronorte-Belém e o fluxograma do software já desenvolvido para supervisão em tempo real do sistema. estão reportados em [1].

3. MONITORAÇÃO DA POTÊNCIA ELÉTRICA DE INTERCÂMBIO

A potência elétrica de intercâmbio entre duas áreas seguem programações pré-estabelecidas pelos setores de planejamento das empresas envolvidas. Com isso. o compromisso em se manter a programação é muito grande pela empresa que fornece a energia. visto que o SE N/Ne opera em determinadas épocas do ano no seu limite de estabilidade conforme citado anteriormente. onde desligamentos de linhas de transmissão. compensadores síncronos. bancos de capacitores. perda de máquina na CHESF e grandes rejeições de blocos de carga introduzem riscos de deligamentos devido a redução da margem de estabilidade. Portanto. torna-se necessário que se implemente funções no sistema de supervisão que aumente a confiabilidade. Para tal é necessário a monitoração em tempo real.

Atualmente os operadores dispõem de instruções operativas contidos em extensas lista de dados contendo as informações necessárias para manter os limites dentro dos planejados. Estas listas mostram várias tabelas de limites que variam em função de:

- Configuração do sistema
- Fluxo em determinada linha
- Posicionamento dos tapes dos ATR's
- Quantidade e disposição de reatores instalados
- Estado de disjuntores essenciais
- Equipamentos de controle de reativo
- Eventos simultâneos

Como se pode notar. a gama de dados a ser tratado é muito grande. Dentre alguns tratamentos sugeridos. a opção adotada foi a do processamento paralelo Para isto procederam-se estudos para separação adequada de dados sem que incorresse em incoerências que comprometessem o novo processo de monitoração.

A configuração do sistema e o fluxo ativo em determinada linha foram as escolhidas para fazer parte do primeiro grupo de dados a ser tratado.

3.1. CONFIGURAÇÃO DO SISTEMA

O sistema Eletronorte é composto basicamente de linhas principais radiais em 500Kv. tendo como fonte principal a usina de Tucuruí composta de 12 máquinas de 330 Mw. como mostrado na figura 1. Todas as subestações de 500 Kv são em arranjo disjuntor e meio [3].

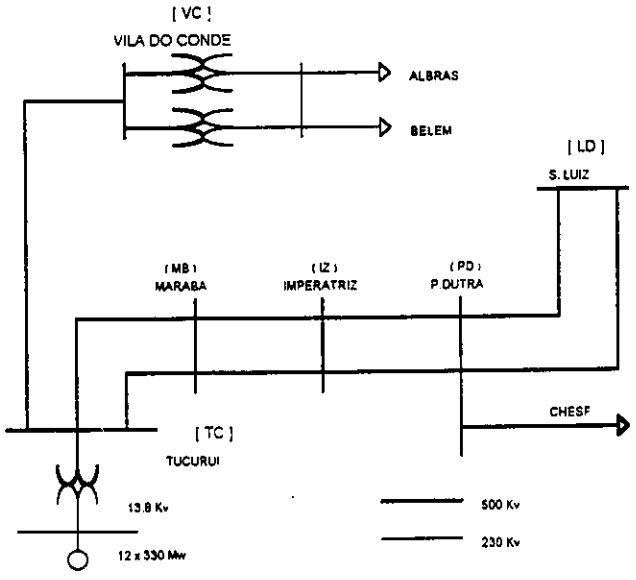


Figura 1. **Sistema Eletronorte**

Cada trecho de linha e composto de 2 seccionadores de linha (SL), 8 seccionadores de disjuntores (SD) e 4 disjuntores (DJ). Estes equipamentos se dividem igualmente para cada lado, como mostra a figura 2

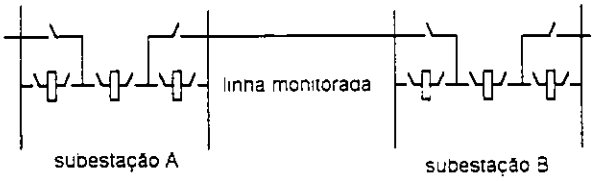


Figura. 2 - **Configuração de cada trecho de linha**

3.2 - LIMITE MÁXIMO DE FLUXO

Baseados em estudos de planejamento da operação determinou-se limites para o fluxo de potência ativa na linha de transmissão (500 Kv) de Tucurui à Maraba em função da indisponibilidade de linhas do sistema como mostrado na tabela 1

Trecho da linha	CIRCUITO I	CIRCUITO II	Fluxo maximo Mw na LT - TC / MB
TC - MB	aberto	fechado	1280
TC - MB	fechado	aberto	1170
MB - IZ	aberto	fechado	1260
MB - IZ	fechado	aberto	1180
IZ - PD	aberto	fechado	1060
IZ - PD	fechado	aberto	1000
PD - LD	aberto	fechado	1130
PD - LD	fechado	aberto	1100
qualquer	aberto	aberto	avisar ao operador

TABELA 1 - **Limites de fluxo de potência na linha TC / MA em função da configuração**

A tabela acima expressa parte da estratégia a ser adotada pelo operador do sistema elétrico da Eletronorte, escolhida para servir de referencial para o primeiro procedimento a ser seguido na monitoração via RNA. Entretanto, ao serem tratados estes dados, leva-se em consideração que as demais partes da estratégia estejam normalizadas.

A fim de reduzir a quantidade de dados a ser tratado de (2SL + 3SD + 4DJ) * 9Linha - $9F_{max}$ = 135 para (4CN - 2SL) * 9Linha - $9F_{max}$ = 63, são utilizadas variáveis calculadas (CN) que fornecem o resumo dos estados logicos de cada conjunto mostrado na figura 3 indicando, se o conjunto esta aberto ou fechado.

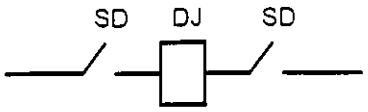


Figura 3 -**Conjunto usado para cálculo de CN**

4. IMPLEMENTAÇÃO DA RNA

O gerenciador de dados [1] da RLS fornece em tempo real o status de todas os equipamentos (SL, CN) necessários para o tratamento. Alem dos eventos citados na tabela 1 incluiu-se a possibilidade de indisponibilidade de um trecho do eixo TC/MB/IZ/PD. Para isto, tomou-se o cuidado de apresentar os dados de forma sequêncial para facilitar o tratamento via software.

Devido ao grande numero de dados a ser tratados, dividiu-se o trabalho neuronal em duas etapas. Criando duas redes neuronais artificiais interligadas serialmente [4 - 7] A primeira RNA - chamada de Rede Neuronal Artificial Primaria (RNAp) - fornece à segunda RNA - chamada de Rede Neuronal Artificial Secundária (RNAs) - dados estrategicos para a análise final. A figura 4 mostra a configuração geral da RNA.

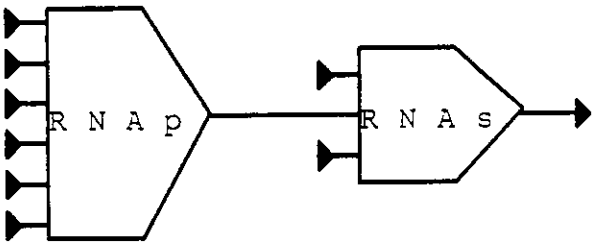


Figura 4 - **Configuração geral da RNA**

4.1 - A RNAp

A RNAp recebe do gerenciador de dados do sistema os 6 estados lógicos referentes a cada circuito de linha, conforme mostrado na figura 5.

Figura 5 - **Estados lógicos referentes a cada linha**

Os estados citados acima servem de entradas para a RNAp que e configurado conforme figura 6. A saida da RNAp que pode ser 0 ou 1 serve de entrada para a RNAs. Dependendo da combinação das entradas pode-se determinar se a linha está fechada ou em aberto

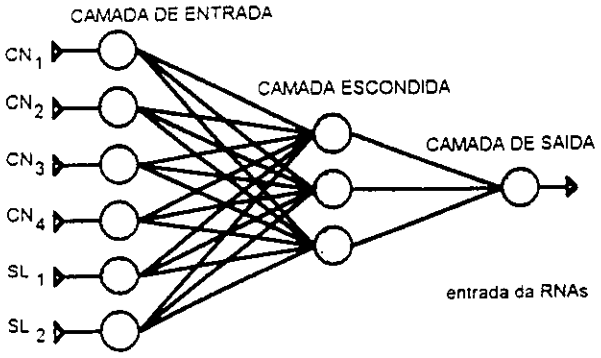


Figura 6 - **A RNAp**

As $2^6 = 64$ combinações de 0 e 1 das chaves CN e SL serviram de entrada e 0 e 1 de saida para o treinamento da RNAp. no qual foi utilizado o método do Back-Error-Propagation

Por serem as condições de configuração do sistema muito bem definidas. pois. as entradas são de 0 e 1. tem-se a saida da RNAp como 0 ou 1. Como a RNAp utilizada possui funções ativação do tipo linear na camada de entrada e sigmoidal nas demais. utilizou-se chaves logicas IF/ELSE para obter a saida desejada.

4.2. A RNAs

Como o objetivo desta implementação é apresentar o fluxo máximo de potência ativa na LT TC/MB aos operadores do sistema. então utilizou-se como saida da RNAs. o fluxo máximo para cada situação. Os elementos de entradas definidos. são mostradas a seguir:

- Entrada 1→Estado do segundo circuito do trecho estudado
- Entrada 2 → Saida da RNAp
- Entrada 3 → Número de ordem da linha / 10

O objetivo da entrada 1 é fornecer meios para se determinar se o trecho da linha monitorada está interrompido ou não. Caso positivo a RNAs deve detectar.

A RNAs possue 3 camadas sendo 3 neurônios na camada de entrada. 5 na camada de escondida e 1 na de saida. como mostra a figura 7. Com excessão da camada de entrada que possui função ativação linear. todos neurônios possuem função ativação sigmoidal sem a utilização de deslocadores de função

No treinamento das RNA's foi utilizado um software desenvolvido na Eletronorte. que permite diversas manobras quanto a mudança de constantes como: taxa de aprendizagem. fator de esquecimento. desvio padrão e valor médio de pesos iniciais. erro máximo permitido. etc Também permite a inclusão ou exclusão de entradas para treinamento.

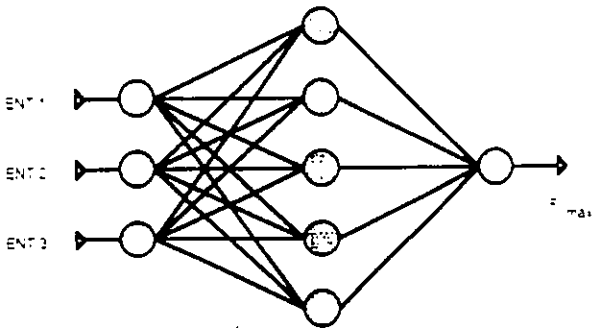


Figura 7 - **A RNAs**

4.3 - O SOFTWARE

No ambiente em que foi desenvolvido o trabalho. ainda é impossível a implementação de RNA via hardware. logo implementou-se via software. O programa foi desenvolvido em liguagem C - Watcom [QNX] que gerencia todo o processo. A figura 8 mostra o fluxograma seguido para se conseguir a análise completa de

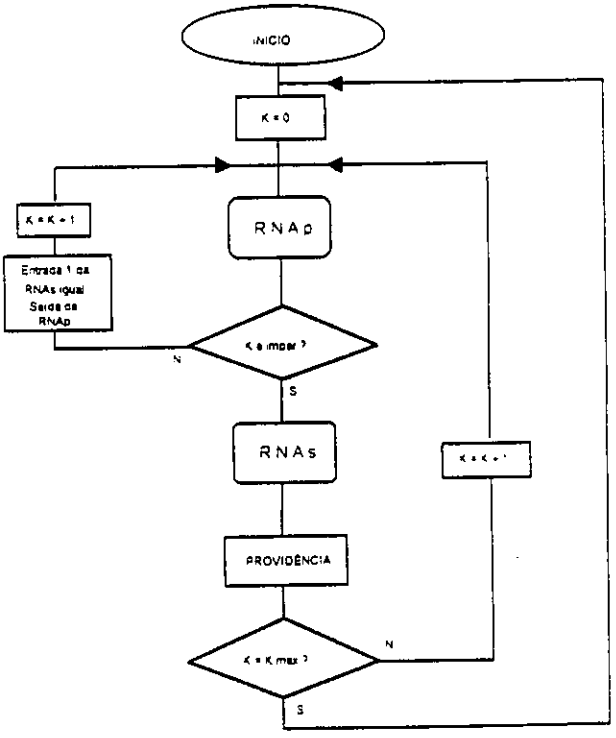


Figura 8 - **Fluxograma do Software desenvolvido**

A constante K que aparece no fluxograma é definida como a ordem de apresentação dos circuitos. Como estão em ordem par a par, pode-se dizer que o circuito anterior ao monitorado quando K for impar, é o circuito par do mesmo.

Como linha TC/VC tem apenas um circuito, definiu-se que ela é par dela mesma, o que não invalida o resultado.

5. CONCLUSÕES

O trabalho apresentou uma implementação em tempo real de uma Rede Neuronal Artificial via "software". Implementação esta permitida devido a utilização de um sistema operacional multi-tarefa, multi-usuário e com priorização de tarefas.

A possibilidade de se poder implementar novas estratégias ou novos tópicos no trabalho é o ponto de maior relevância quanto a continuidade do trabalho no tratamento de novos dados. Sem que haja mudanças significativas nos software's desenvolvidos, é possível incrementar dados, pois, para tal pode-se tomar várias medidas que trariam resultados satisfatórios.

A mudança de maior flexibilidade seria nos pares de entrada/saida para treinamento da RNAs. Com a inclusão de novas entradas pode-se aumentar o nível de monitoração e com novas saídas aumentar o diagnóstico. No futuro pretende-se incluir na RNAp circuitos fictícios que exprimem o estado de conjuntos de equipamentos de controle de reativos e na saída da RNAs tensões máximas e mínimas nas barras de 500 Kv.

Outra implementação significativa seria a utilização do mesmo programa, com algumas modificações de código, para serem processados paralelamente no mesmo ambiente onde foi desenvolvido este trabalho.

6- REFERÊNCIAS

[1] - BRITO, P. R; GARCEZ, J. N; CHARONE, W. : "Um Programa para Assistência ao Controle de Geração da Eletronorte Usando Rede Neuronal Artificial". Anais do IV SEPOPE - Brasil: pp SP-44, maio 1994.

[2] - **QNX Software Systems Ltd.**
Kanata, Ontario K2M 1W8 Canada.
Email: **postmaster@quantum.qnx.com**

[3] - Relatórios Técnicos da Eletronorte.

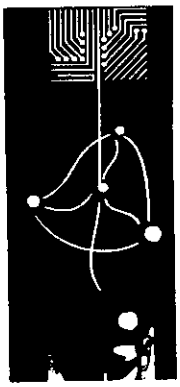
[4] - SIMPSON, P. K. - "Artificial Neural Systems"- Pergamon Press - 1990.

[5] - HUNT, K.J; SBARBARO D; ZBIKOWSKI, R; GAWTH-ROP, P.J; Neural Networks for Control Systems - A Survey - Automática, Vol 28, no. 06 pp: 1083 - 1112; 1992.

[6] - PASSINO, K. M; (1993) - Bridging the Gap Between Convencional and Intelligent Control - IEEE - Control Syst. Mag. pp 12-18.

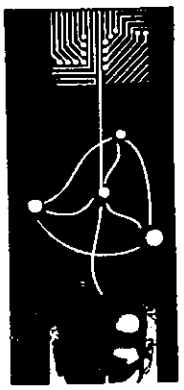
[7] - ANTSAKLIS, P.J. PASSINO, K.M; (1993). An Introdução to Intelligent and and Autonomous Control - Nor-well, MA: kluwer.

Anotações



NEUROCIÊNCIA E BIOMÉDICA

Anotações



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

APLICAÇÃO DE REDES NEURAIS À AVALIAÇÃO DO DESEMPENHO DE SISTEMAS CARDIOVASCULARES

EDSON PACHECO PALADINI
DEPTO. DE ENGENHARIA DE PRODUÇÃO E SISTEMAS
UNIVERSIDADE FEDERAL DE SANTA CATARINA

RESUMO

O presente trabalho desenvolve uma aplicação de Redes Neurais para o diagnóstico de sistemas cardiovasculares. O processo utilizado consiste na análise de imagens obtidas do paciente em observação. Resultados preliminares mostram que a rede projetada para resolver o problema atende adequadamente ao que requer-se de diagnósticos desta natureza.

1. INTRODUÇÃO

O século XX tem assistido a um considerável incremento no conhecimento que a ciência médica tem desenvolvido a respeito do funcionamento do sistema cardiovascular e dos males mais usuais que o envolvem. Este desenvolvimento está igado à estruturação de métodos avançados de diagnóstico cardíaco, um processo sempre complexo. As pesquisas nesta área hoje não mais apontam para ações que tentam classificar um paciente em algum tipo de doenças conhecidas: os médicos modernos desenvolvem uma avaliação exaustiva e objetiva da anatomia, fisiologia, prognose e do planejamento do tratamento (Skorton et.al., 1991:1).

O processo do diagnóstico ainda começa com a obtenção do histórico do paciente e o desenvolvimento de exames físicos, usualmente acompanhados de eletrocardiogramas e raios-X do tórax. Baseado nestas informações iniciais, o médico faz um diagnóstico preliminar e formula hipóteses, cuja confirmação e extensão dependem de uma bateria de ferramentas básicas de diagnóstico, que incluem análise bioquímica do sangue e urina; registros de eletrocardiograma feitos com o paciente em repouso, e, a seguir, submetido a exercícios graduais, bem como o estudo das cavidades cardíacas. Tendem a acompanhar esta avaliação a análise de imagens do coração e do sistema de vascularização, obtidas por sofisticados métodos como a ecocardiografia. Com exceção da eletrocardiografia e dos testes eletrofisiológicos, os procedimentos mais comuns para a execução do diagnóstico estão relacionados com os métodos de obtenção de imagens. Assim, a captura de imagens cardíacas tem se tornado no mais importante elemento na formulação de modernos diagnósticos cardiovasculares (Skorton, et al., 1991: 2).

2. PROCESSOS DE CAPTURA E ANÁLISE DE IMAGENS CARDÍACAS

Os métodos de captura de imagens do coração e das áreas que o cercam permitem que o médico visualize a anatomia cardíaca e determinam elementos quantitativos do sistema cardíaco, como a determinação do tamanho, forma e estrutura das cavidades cardíacas (ventrículos e átrios), o diâmetro das válvulas e dos folículos, bem como as artérias coronárias. A seguir, atenta-se para as funções valvulares e das cavidades, incluindo a avaliação das funções sistólicas e diastólicas com o estudo da severidade da estenose valvular (regurgitação).

Até recentemente, os métodos de produção de imagens cardíacas estavam restritas a estas duas áreas - anatomia e funções. Na década de 80, progressos em laboratório e processos experimentais foram efetivados, realçando três novos objetivos de utilidade clínica indiscutível: a perfusão, metabolismo e características básicas do miocárdio (Skorton,D. & Collins, S., 1985:803). Hoje, quando enfatiza-se a minimização de efeitos graves causados pela isquemia aguda do miocárdio (TIMI, 1989:690), os médicos desejam informações precisas relativas à perfusão do miocárdio. Idealmente, esta informação incluiria estimativas dos níveis de perfusão do paciente em repouso, submetido a tratamento farmacológico ou de stress físico e durante os processos de isquemia aguda (Marcus, M. et al., 1987:97). Em um nível mais básico, tem-se procurado determinar aspectos relevantes ligados aos problemas cardíacos no metabolismo do miocárdio (Grover, 1986:290). Finalmente, os médicos buscam informações referentes à composição do tecido miocardial, o que inclui a identificação de deposição anormal de colágenos, ferro, amilóides ou outras substâncias, delineamento da natureza das massas intracardíacas e identificação dos miocárdios cronicamente isquêmicos.

Estes aspectos mostram que os objetivos da captura e análise das imagens cardíacas é amplo e a relevância que a ciência médica confere à avaliação destas imagens, pela diversidade de análises a fazer e pela complexidade dos descritores do sistema (Skorton, et al., 1991:4). Esta relevância tem contribuído para o desenvolvimento de técnicas quantitativas com tal finalidade e produzido inúmeros trabalhos nesta área (ver, por exemplo, Collins, et al., 1988). Em particular, na área de processamento de imagens, existem estudos que analisam níveis de cinza de imagens monocromáticas de áreas específicas do sistema cardiovascular e já se analisam imagens digitalizadas do coração através de funções contínuas ou de matrizes de representação dos níveis de cinza (Feagle, S. & Skorton, D., 1991:72).

3. APLICAÇÃO DO MODELO AOS PROCEDIMENTOS DE AVALIAÇÃO DO SISTEMA CARDIOVASCULAR

O modelo de análise aqui proposto envolve a utilização de uma rede neural e de um modelo posterior de Reconhecimento Analítico de Padrões. A idéia do processo é utilizar uma "imagem original", que seria o padrão, e um conjunto de imagens sucessivas, que mostram a evolução do quadro clínico do paciente. Isto mostra a natureza dinâmica do processo. O uso do modelo pode conferir uma base objetiva e quantitativa para esta análise, o que auxilia grandemente o diagnóstico a ser processado e minimiza muitos dos problema que a chamada avaliação "perceptual", hoje largamente empregada pelos médicos, ainda traz. Considera-se, em geral, que os momentos mais cruciais do julgamento (que determina a formulação do diagnóstico) ocorre na percepção de imagens que representam situações de contínuo movimento (Franken, E. A. & Berbaum, K. S., 1991:88). Esta situação é aqui adequadamente tratada.

Em função das características do modelo e do problema a ser considerado, a primeira aplicação proposta (que ora se desenvolve) diz respeito, especificamente, à avaliação dos elementos determinantes da função ventricular sistólica e da função ventricular diastólica.

Em linhas gerais, esta é a motivação da aplicação em questão: A precisão das informações acerca das funções ventriculares é essencial para a avaliação de pacientes com doenças cardíacas. Compreende-se, assim, porque a maioria das técnicas que envolvem imagens dos sistemas cardiovasculares hoje em uso tenha, entre seus objetivos principais, a análise da estrutura dos ventrículos e das funções sistólica e diastólica por eles desempenhada (Marcus, M. & Dellsperger, K. C., 1991:24).

O status funcional e as características anatômicas dos ventrículos são importantes em muitos aspectos. Em primeiro lugar, o prognóstico para um paciente com quase qualquer tipo de doença cardíaca (disfunções coronarianas, disfunções valvulares, cardiomiopatia e problemas devidos à hipertensão do coração) é influenciado pela situação atual do desempenho dos ventrículos. Note-se que a taxa anual de mortalidade de pacientes por imperfeições das funções do ventrículo esquerdo chega a 20% (Mock, 1992:562). Em segundo lugar, o mecanismo responsável por sintomas clínicos e sinais de falhas cardíacas por congestão pode ser uma disfunção de um ou de ambos os ventrículos, particularmente em termos dos movimentos sistólico ou diastólico (ou ambos). Pesquisa feita nos Estados Unidos mostra que, em 30% dos adultos com problemas cardíacos, a origem da doença está em alterações da função diastólica do ventrículo esquerdo (Braunwald, 1988: 471).

Por fim, certas falhas funcionais ou especificidades de cada um dos ventrículos do paciente são essenciais para o correto diagnóstico do problema cardíaco que o acomete, como o aneurisma ventricular e vários tipos de estenoses (Marcus, M. & Dellsperger, M., 1991:25). Assim, a informação precisa sobre a estrutura e a função dos ventrículos é fundamental para o diagnóstico, já que pode, com frequência, identificar a causa precisa da doença do paciente.

Se a questão acima não parece suscitar dúvidas entre os médicos envolvidos com pacientes cardíacos, já a definição da natureza da precisão da informação é uma questão que desperta controvérsias. O que se sabe, concretamente, é que a forma dos elementos cardíacos, os movimentos dos ventrículos (que são diferentes de um para o outro) e a anatomia ventricular são aspectos a considerar em qualquer avaliação. Uma análise precisa de imagens que represente tais elementos, assim, é de importância indiscutível para o tratamento do paciente.

Muitos destes aspectos podem ser tratados no âmbito do modelo aqui proposto. Considere-se um pequeno exemplo: O movimento do ventrículo esquerdo, durante a sístole, envolve cinco movimentos específicos: (1) translação; (2) rotação; (3) "torcedura"; (4) movimento tipo sanfona e (5) movimento da circunferência endocardial em torno do centro da cavidade ventricular. Estes movimentos não são uniformes. Por exemplo, o movimento tipo sanfona é extremamente assimétrico.

Os dois movimentos mais importantes são o da superfície endocardial (5) e o tipo sanfona (4). Eles podem influenciar todo o funcionamento do ventrículo (Chairtman, 1973:1043). Eles são detectáveis por ecocardiografia bidimensional, ventriculografia com contraste ou tomografia computadorizada, entre outros métodos.

A ocorrência de problemas cardíacos pode, com frequência, causar um aumento desproporcional de um ou mais componentes do movimento do ventrículo esquerdo. Nenhum método (dentre aqueles em uso hoje) pode medir a contribuição específica de cada um dos movimentos primários do ventrículo esquerdo; nenhum, por exemplo, capta o movimento de torcedura (Marcus, M. & Dellsperger, M. D., 1991:26). A prática clínica, entretanto, raramente considera os três primeiros movimentos; já praticamente todos os métodos enfatizam os dois últimos.

Desta forma, parece ser relevante considerar dois problemas específicos aqui: como concentrar a análise em mais de um movimento e como avaliá-los. Uma primeira maneira de enfrentar o problema consiste em dispor de várias imagens da área ventricular esquerda, e, a seguir, associar a elas matrizes de níveis de cinza. Segundo alguns testes experimentais desenvolvidos, pode-se proceder da seguinte forma: inicialmente aplica-se, às imagens, um estudo baseado na rede neural proposta. O resultado da aplicação da rede classifica as imagens em grupos que apresentam valores adequados a um parâmetro. A seguir, considerando-se que pode-se tratar de imagens de alguma forma similares, aplica-se, a cada grupo, um segundo modelo, analítico, de Reconhecimento de Padrões, que define intervalos dentro dos quais as imagens estão variando. Por fim, um terceiro modelo, também analítico, é utilizado para determinar áreas específicas de variações relevantes.

Um caso prático é útil para ilustrar esta aplicação. Um paciente chega ao médico queixando-se de dores no peito, febres constantes e

pequenos problemas respiratórios. O médico, então, utiliza-se de um roteiro de perguntas pré-elaboradas e, através das respostas, suspeita da ocorrência de massas intracardíacas, como algum tipo de tumor.

São obtidas, após o questionamento, imagens do coração do paciente em duas posições: a primeira, seguindo o grande eixo, mostra os lados esquerdo e direito do coração (ressaltando o ventrículo e o átrio de cada lado); a segunda, mostra as quatro cavidades (imagem frontal). Como não há protuberâncias à vista, o paciente é liberado, devendo voltar 10 dias mais tarde para novos exames.

Transcorrido um mês, o paciente retorna queixando-se de problemas respiratórios mais agudos. Foram feitas novas imagens e que passaram a ser acompanhadas com mais atenção. Na análise das imagens do movimento de sístole, a aplicação da rede neural identificou variações inaceitáveis; estes valores foram mais significativos no movimento de diástole. Assim, pode-se concluir que, pela posição das alterações observadas, o problema devia estar entre os ventrículos, em particular, no esquerdo. O paciente foi internado, e foram obtidas novas imagens.

Digitalizada em ambiente monocromático, com 16 níveis de cinza, estas imagens possibilitaram que a rede identificasse várias alterações. O uso de um modelo que utiliza distância entre matrizes delimitou as áreas com maiores variações. Novas imagens analisadas pela rede neural mostraram a evolução da massa intra-cardíaca (variações mais aceituadas) 5 dias após a internação do paciente, confrontando apenas o movimento de diástole. Estas informações permitiram confirmar o diagnóstico: o paciente porta um tumor cardíaco, conhecido como mixoma, aparecendo no ventrículo esquerdo durante a diástole e restrito ao átrio esquerdo durante a sístole. Trata-se de um tumor benigno; entretanto, pelo porte do tumor no caso do paciente em questão, o procedimento requerido é cirurgia imediata.

A partir de exemplos como o acima listado, foram simuladas situações adequadas ao problema e encontrou-se alguns resultados satisfatórios. A aplicação, porém, foi prejudicada por um problema específico: a falta de um dispositivo de digitalização adequado a imagens produzidas em vídeo ou fotos pouco nítidas. A imersão do sistema em meio denso trouxe, também, diversos problemas para a captura da imagem.

4. PROJETO DA REDE

A partir da análise processada em um conjunto de imagens, cuja avaliação da variabilidade é conhecida, o programa, com a utilização de uma rede neural, determina se uma imagem particular apresenta, para suas características básicas, variações compatíveis com seu "projeto" original.

Como se tratam de análises monocromáticas, o programa avalia tais variações através dos níveis de cinza da imagem em estudo, fornecidas à rede na fase de treinamento. Serão consideradas "conforme os padrões" as imagens que apresentarem valores compatíveis de variações com as imagens utilizadas para treinamento. Os níveis de cinza das imagens são representados sob forma matricial. Para a operação do programa, os agentes de decisão devem selecionar imagens que são julgadas, a priori, como portadoras de variações aceitáveis e as que mostram variações incompatíveis com a natureza do processo clínico, evidenciando as diferenças entre elas.

Existem várias técnicas para a montagem de uma rede neural. No caso da atual aplicação, observou-se grande facilidade de treinamento, classificação de itens como forma básica de operação e convergência para valores específicos. Além disso, a rede deve trabalhar no sentido de minimizar seus erros, enfatizando as características que possam conduzir a uma decisão correta quanto às variações observadas. Para tanto, torna-se importante criar dispositivos favoráveis à minimização de desvios. Assim, deve ser estruturado um modelo com forte e permanente realimentação, de forma que, sempre que se avance no sentido da obtenção do resultado da decisão, possa-se retornar às informações básicas para checar a validade da operação de toda a rede e, em particular, de decisões específicas obtidas - ou seja, permanente atualização de dados. Decorre daí a sugestão de uso de uma rede "backpropagation".

Em termos conceituais, a rede utiliza a técnica do "gradiente descendente" (Lawrence, 1992:239), adequada para casos onde se busca o tipo de minimização de erros como a descrita acima. A rede estruturada tem 3 camadas,

sendo a primeira representada pelos dados de entrada X(i), ou seja, valores da propriedade da imagem associadas a cada pixel i. Anota-se a saída da rede por O(j) e constrói-se, ainda, uma camada intermediária H(k). A camada de entrada está ligada à camada intermediária por pesos w(i,j), do ponto de entrada i para o ponto da camada intermediária j. Daí para a camada de saída, são utilizados pesos v(j,k). Foram adicionadas duas unidades-fantasmas para limiares, que originam as ligações extras w(n + 1,j) e v(m+ 1,k).

Os valores dos pesos w e v são inicializados por uma subrotina, ALEAT, que gera números aleatórios entre -0.01 e 0.01. Foram fixados os seguintes valores básicos x(n + 1) = 1.0 e h(m + 1) = 1.0

A seguir, inicia-se a fase de treinamento, onde pares de entradas e saídas (x,y) são apresentados à rede, ou seja, x representa o vetor de entradas e y representa o vetor das saídas desejadas. Utiliza-se uma função de ativação exponencial (Knight,1990:68):

$$h(j)=1/(1+\exp(-\sum_{i=1}^{n+1} [w(i,j).x(i)])$$ (j varia de 1 a m; S é o somatório).

Por sua vez, da camada intermediária para a camada final tem-se:

$$o(j) = 1/(1 + \exp(-\sum_{i=1}^{m+1} [v(i,j).h(i)]),$$ j varia de 1 a t.

Obtém-se, assim, a saída proposta pela rede e pode-se definir uma estratégia para minimizar o erro entre a saída proposta pela rede(o) e a saída desejada (y). Utilizando a técnica do gradiente e a regra da cadeia à função de ativação tem-se:

$d(j) = o(j)(1 - o(j))(y(j) - o(j))$ (j varia de 1 a t).

Pela "backpropagation":

$$f(j) = h(j)(1 - h(j))(\sum_{i=1}^{n} [d(i).(v(j,i)])$$ (j varia de 1 a m)

O ajuste de pesos é feito da camada intermediária para a camada final ($A(v(i,j)) = q . d(j) . h(i)$, i varia de 1 a m + 1 e j varia de 1 a n) e da camada inicial para a camada intermediária ($A(w(i,j)) = q . f(j) . x(i)$, i varia de 1 a m + 1 e j varia de 1 a n). q é a a taxa de aprendizagem e seu valor foi fixado em 0.35 (Knight, 1990:68).

O valor do erro a cada interação diminui se as imagens apresentam variações aceitáveis. Quando tornam-se inferiores a um dado limite, a rede está treinada, e os pesos são adequados para a classificação desejada. Parte-se, então, para o processo de avaliação das imagens.

8. CONCLUSÕES

Nos testes do modelo foram processadas cerca de 15.000 imagens, para cerca de 300 situações. Foram observados dois tipos de erros: (1) imagens portadoras de variações aceitáveis classificadas como portadoras de variações relevantes e (2) imagens inadequadas classificadas como perfeitas. Em 3000 imagens utilizadas como amostra, 99.85% foram corretamente classificadas, havendo erro em apenas 0.15% delas.

Em geral, observa-se um bom desempenho do programa, embora considere-se uma restrição básica de operação: não são feitas análises detalhadas dos pixels, mas apenas uma avaliação global da imagem. Pelo uso de um modelo baseado em distâncias, porém, pode-se localizar áreas problemáticas e desenvolver-se um diagnóstico mais preciso.

REFERÊNCIAS BIBLIOGRÁFICAS

Braunwald, E. Clinical manifestations of heart failure. In: Heart Disease. Philadelphia, Saunders Company, 1988.

Chairtman, B. Left ventricular wall motion assessed by using a fixed external reference system. Circulation, Vol. 48. (1043-50). 1973.

Collins, S. M. and Skorton, D. J. (Eds.) Cardiac Imaging and Image Processing. New York, McGraw-Hill Book Company, 1986.

Feagle, S. R. & Skorton, D. J. Quantitative Methods in Cardiac Imaging. in: Cardiac Imaging. London, Saunders Co, 1991.

Franken, E. A. & Berbaum, K. S. Perceptual Aspects of Cardiac Imaging. in: Cardiac Imaging. London, Saunders Co., 1991.

Grover, M. K. Identification of impaired metabolic reserve in patients with significant coronary artery stenosis by atrial pacing. Circulation, Vol. 74. Págs. 281-300. 1986.

Knight, K. Connectionist Ideas And Algorithms. Communications of the ACM. Vol. 33, N. 11. 1990. P. 59-74.

Lawrence, J. (Ed.) Introduction to Neural Networks and Expert Systems. Nevada City, CA. California Scientific Software. 1992.

Marcus, M. L.; Wilson, R. F. & White, C. W. Methods of measurement of myocardial blood flow in patients. Ciculation, Vol. 76. (245). 1987.

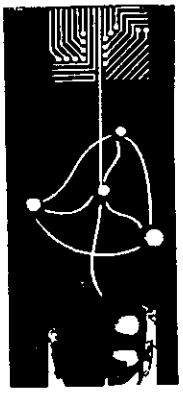
Marcus, M. & Dellsperger, K. C. Determinants of Systolic and Diastolic Ventricular Function. in: Cardiac Imaging. London, Saunders, 1991.

Mock, M. B. Survival of medically treated patients in the coronary artery surgery studay registry. Circulation, Vol. 66. (562-590) 1992.

Skorton, D. J.; Marcus, M. L.; Schelbert, H. R. & Wolf, G. L. Cardiac Imaging. Philadelphia, W. B. Saunders Co., 1991.

Skorton, F. J. & Collins, S. M. New directions in cardiac imaging. Annual International Medical. Vol. 102, Págs. 795-840, 1985.

TIMI Study Group: Comparison of invasive and conservative strategies after treatment with intravenous tissue plasminogen activator in acute myocardial Infarction. New England Journal of Medicine. Vol. Vol. 320. Págs. 618-700, 1989.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá. 24 a 27 de outubro de 1994

AUTOMATIC DETECTION OF SLEEP-WAKING STATES USING KOHONEN NEURAL NETWORKS.

COIMBRA, A.J.F.*; MARINO-NETO, J.†, FREITAS, C.G.†; DE AZEVEDO, F.M.*; BARRETO, J.M.‡

* Grupo de Pesquisas em Engenharia Biomédica (GPEB), EEL, Universidade Federal de Santa Catarina, Brazil.
† Laboratório de Neurofisiologia I, CFS, Universidade Federal de Santa Catarina, Brazil.
‡ Neurophysiology Laboratory, Université Catholique de Louvain, Belgium.

ABSTRACT

The present work describes the development and preliminary tests of a method for sleep-waking electrographic signal analysis and pattern recognition using self-organizing Kohonen's Artificial Neural Network (KNN) topology.

INTRODUCTION

In recent years, neural networks (NNs) have been proposed as useful tools for tackling analytical and classification problems in an increasingly high number of biological areas. (anesthesiology, EEG classification and wave-form detection, regional blood flow estimation, diagnosis, see, e.g. Bankman et al, 1992; de Azevedo, 1993; Garcia, 1992; Mamelak et al, 1991; Wu and Slater, 1993). The general advantages of NNs for biological signal processing include high tolerance in dealing with noisy, multimodal, incomplete and/or contradictory data, while extracting and recognizing patterns and intrinsic characters from complex interactive processes such as the biological ones. By far, the most promising biological field for NNs is neurobiology, particularly in processing brain generated electrical signals (Bankman, 1992; Klöppel, 1994a,b; Mamelak et al, 1991; Roberts and Tarassenko, 1992; Schaltenbrand et al, 1993). In fact, the benefits of such an interaction are not confined to biology. The development of NNs methods for complex, "real world", data (like electroencephalographic ones) are supposed to bring improvements to NNs performances with other kind of data (Klöppel, 1994a)

Electrographic sleep signals analysis takes into account a number of interacting phenomena, including electroencephalographic (EEG, electrical oscillations recorded at brain or scalp surface), electromiographic (EMG, electrical oscillations due to muscle activity), electrooculographic (EOG, electrical oscillations due to eyeball movements), and systemic/vegetative data, as arterial pressure, cardio-respiratory activity, metabolism and temperature. Sets and oscillations of all these attributes define a number of sleep-waking stages (or states), and can be used for research and diagnostic purposes for some sleep and mood disorders (Carskadon and Dement, 1988; Reimão, 1990; Timsit-Berthier, 1990).

The present work describes the development and preliminary tests of a method for sleep-waking electrographic signal analysis and pattern recognition

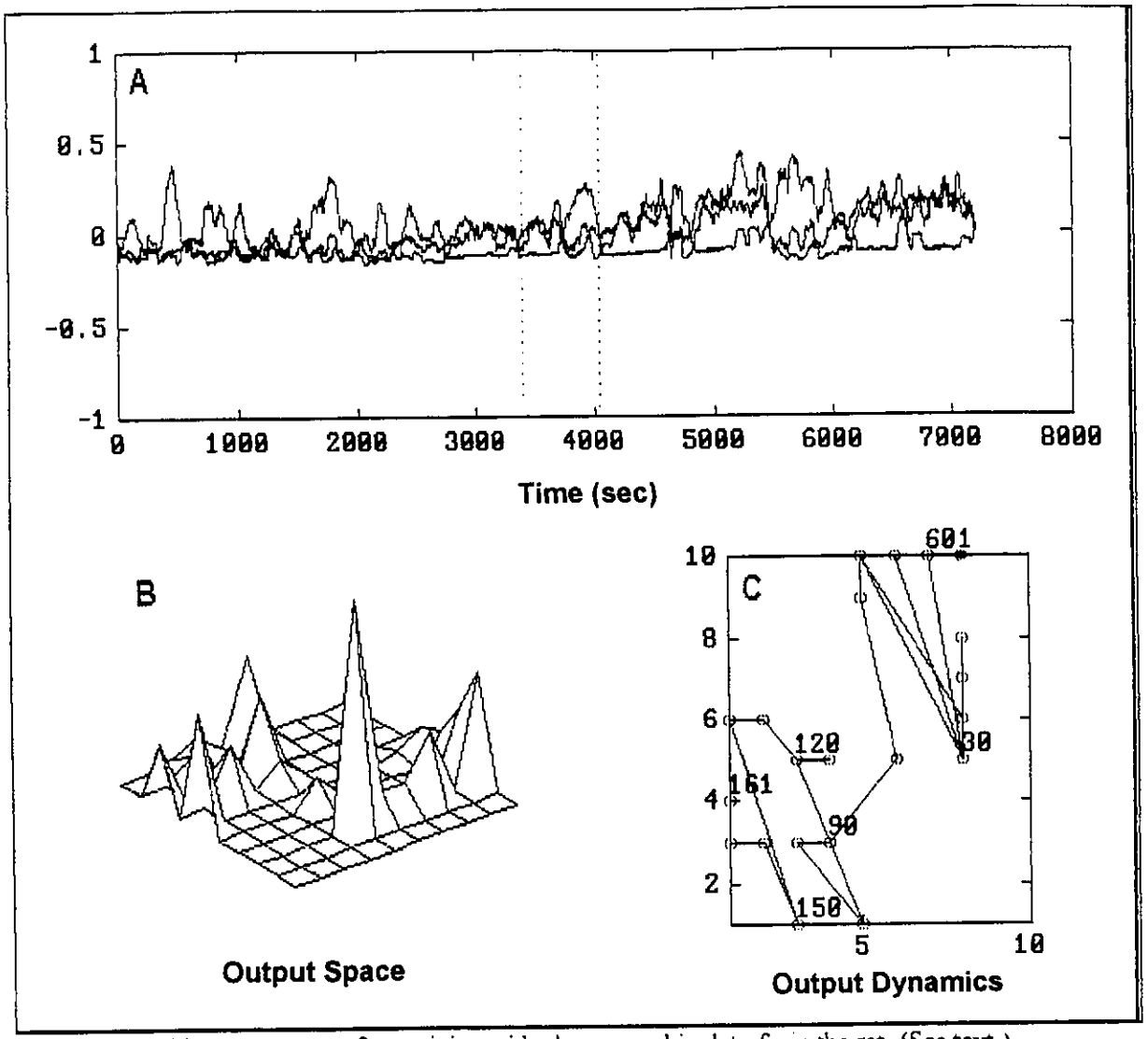


Figure 1 - KNN output space after training with electrographic data from the rat. (See text.)

using Kohonen's Artificial Neural Network (KNN) topology. The use of KNN in this study aims first at the automatic classification of the sleep-waking stages. These procedures are intended to abbreviate and to add accuracy to the laborious and time-consuming routine analysis tasks performed by EEG sleep experts in clinical and research activities. Furthermore, the present work is part of a research effort developed at our labs, designed to study the potential of NN methods for the automatic analysis of biopotentials of clinical relevance.

Most of the recent attempts in using of NNs for the above mentioned purposes employ topologies with supervised learning algorithms (Klöppel, 1994a,b; Mamelak et al, 1991; Schaltenbrand et al, 1993). Such an approach can bring about a number of problems in EEG analysis, including the existence of criteria disagreements among experts and different classifying rules for different abnormal sleep patterns. Roberts and Tarassenko (1992) have recently proposed the use of unsupervised learning algorithms to overcome these problems. Unfortunately, however these authors have employed Kalman filtering to extract the coefficients of the auto-regressive (AR) model of the signal, as input parameters to the KNN. Such parameters, however, are not familiar to the EEG or sleep expert. The use of frequency and amplitude components of the signals is more common in routine analysis, and could be more promptly

realized by clinicians/researchers in sleep analysis. We employed here these more familiar signal features to examine their suitability for NN discriminative tasks.

METHODOLOGY

The method developed here is based on the analysis of the dynamics of parameters extracted from the bioelectrical potentials (EEG, EMG, EOG) related to the sleep-waking states in laboratory albino rats, chronically implanted with recording electrodes placed at frontal and occipital cortical surfaces (for EEG), neck muscles (for EMG) and periorbital bones (for EOG). These signals were continuously recorded from these unrestrained rats for as long as 8 hours, during the light phase of the day, so to record samples from most of the sleep-waking continuum.

The signals were first conditioned by a conventional polygraph (amplified and filtered – 2nd order Butterworth low-pass linear phase filter, 30 Hz cut-off frequency, and 0.3 sec time constant), digitized using a 12 bit analog to digital converter, sampled at 64 Hz and stored on an IBM compatible microcomputer hard disk for off-line processing. A software was specifically developed to control the calibration, acquisition and storage processes as well as the analytical procedures (Coimbra, 1994). The electrographic records were partitioned in 4-second epochs (time frames). From the raw data of each epoch 11 numerical parameters were extracted: powers of 4 EEG frequency bands (0.5-30.0, 0.5-4.0, 6.5-9.0, and 12.0-30.0 Hz) and centroid frequency of each channel, and the EMG integral. The time series corresponding to the sequence of values for each parameter was normalized and a 12th parameter was calculated as a linear combination of the other 11, so to form 12 element vectors with norm 1. These formed the input vectors for a self-organizing KNN array (10x10) of processing elements fully interconnected.

Training was performed through the standard Kohonen algorithm (Kohonen, 1984), with a set composed of the 2 first hours of the record. First the connection weights were setup random values. Afterwards, each input vector was randomly picked up from the 1800 vectors pool and fed into the KNN for the updating of the connections weights; this training procedure being repeatedly carried out. After the training phase some "patches" of variable durations were chosen from the records to evaluate the network behavior.

RESULTS & DISCUSSION

Figure 1.A depicts moving averages of only 3 of the 12 parameters measured along an 8-hour record: the EMG module integral, and the powers of 0.5-4.0 and 6.5-9.0 Hz of the frontal EEG. These, and the other 9 parameters, occurring at the time interval defined between the vertical dotted lines, were used in this example as input to the trained KNN, so to produce a 2-dimensional matrix (figure 1.B) illustrating the clusters (regions) of the output space of the KNN activated in that interval. Each of the mesh vertexes represents a processing unit, and the amplitude of each peak represents the unit activation frequency, at the period analyzed. Figure 1.C illustrates the output for the same recording period in a dynamic way indicating the sequence of transitions among the different clusters of output units. In both output space and sequence representations, at least three different clusters can be readily discriminated and temporally followed. These clusters closely correspond to the visually diagnosed states of waking, slow-wave sleep, and paradoxical sleep. These results indicate the suitability and sensibility of the KNN topology, the unsupervised algorithm and the parameters used for

training and classification tasks in discriminating among different sleep-waking states.

It is worth noting that the unsupervised nature of the training algorithm frees the method from any *a priori* (and thus possibly biased) information on these complex parameters sets, so it can also detect transitional and/or unusual association patterns. Handling these data in the sense of analyzing the KNN's feature map, one can also search for feature combinations associated to output events of this kind. Used in this way, such an approach can be employed not only as an automated state detector, but also as a tool for neurological and neurophysiological investigation, supporting the exploration and identification of novel dynamic characteristics of the sleep-waking cycle, and of other behavioral/physiological states where electrographic examination holds.

Additional experiments, using the present methodology, are being carried at our labs, assessing the behavior and suitability of different parameter sets for sleep-waking stage detection and in analysis of pharmacologically-induced changes in ongoing EEG. At present, however, it is reasonable to believe that further developments of self-organizing NN approaches would represent major improvements to the diagnosis and investigation in biological aspects of clinical relevance.

REFERENCES

BANKMAN, I.N., SIGILLITO, V.G., WISE, R.A., SMITH, P.L. Feature-based detection of the K-Complex wave in the human electroencephalogram using neural networks. IEEE Transactions on Biomedical Engineering, v. 39, n. 12, p. 1305-1310, 1992.

CARSKADON, M.A., DEMENT, W.C. Normal human sleep: an overview. In: KRYGER, M.H. (Ed.) Principles and Practice of Sleep Medicine. New York: Saunders, 1988, p. 3-13.

COIMBRA, A.J.F. Análise computadorizada de sinais bioelétricos. MSc Dissertation, 1994, Centro Tecnológico, UFSC, Brasil.

DE AZEVEDO, F.M. Contribution to the study of neural networks in dynamical expert systems. PhD Thesis, 1993, Institut d'Informatique, FUNDP, Belgium.

GARCIA, R.O. Técnicas de inteligência artificial aplicadas ao apoio à decisão médica na especialidade de anestesiologia. PhD Thesis, 1992, Centro Tecnológico, UFSC, Brasil.

KLÖPPEL, B. Application of neural networks for EEG analysis. Neuropsychobiology, n. 29, p. 39-46, 1994a.

KLÖPPEL, B. Classification by neural networks of evoked potentials. Neuropsychobiology, n. 29, p. 39-46, 1994b.

KOHONEN, T. Self-Organization and Associative Memory. Berlin: Springer-Verlag, 1984, 255 p.

MAMELAK, A.N., QUATTROCHI, J.J., HOBSON, A. Automated staging of sleep in cats using neural networks. Electroencephalography and clinical Neurophysiology, n. 79, p. 52-61, 1991.

REIMÃO, R. Sono: Aspectos atuais. São Paulo: Atheneu (Série Neurológica, Psiquiatria), 1990, 300 p.

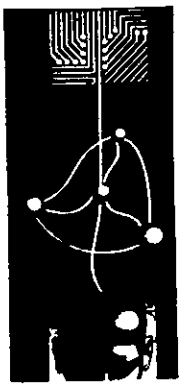
ROBERTS, S., TARASSENKO, L. New method of automated sleep quantification. Medical & Biological Engineering & Computing, n. 30, p. 509-517, 1992.

SCHALTENBRAND, N., LENGELLE, R., MACHER, J.-P. Neural networks model: Application to automatic analysis of human sleep. Computers

and Biomedical Research, n. 26, p. 157-171, 1993.

TIMSIT-BERTHIER, M. Approche neurophysiologique des états dépressifs. Psychologie Medicale, v. 22, n. 8, p. 757-763, 1990.

WU, F.Y., SLATER, J.D. Regional cerebral blood flow estimation by neural network-based parametric regression analysis. Int Journal of Biomedical Computation, n. 33, p. 119-128, 1993.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

Defining Cognition: Adaption by the Unstable Search for Coherent Conservation

Sérgio U. Dani[1,2] and
Gerhard F. Walter[2]

Abstract

Knowledge in the cognitive sciences is becoming more and more holistic, as opposed to fragmentary, in order to be real. An encompassing definition of cognition is then necessary. Evidence is reviewed here which allows to the conclusion that cognition stands for an adaptive ability to environmental changes, operating through non-linear search for coherent conservation of complex organismal features.

1 Departamento de Neurobiologia, Instituto de Biofísica Carlos Chagas Filho, UFRJ, Rio de Janeiro, RJ, Brazil

2 Institut für Neuropathologie der Medizinischen Hochschule Hannover, 30625 Hannover, Germany (correspondence address)

Cognitive science has witnessed a great advancement which has much to do with its multidisciplinary character. As a new branch of science, however, it still deserves the aims of classification. More specifically, a generally accepted lexicon is desirable in order to avoid misconceptions or misunderstandings, to enable easy information exchange and also as a guide to address experimental approaches.

For the sake of achieving these aims, it is worthwhile to make use of different tools from distinct sciences - including those that are usually maintained apart from each other. Chagas (1990) pointed out that a particular science can only be regarded as autonomous just in those respects in which less progress has been achieved. The reason for that, according to Chagas' view, has to do with the realization that knowledge is becoming more and more holistic, as opposed to fragmentary, in order to be real.

To date, some definitions of cognition or cognitive systems, such as "a device to organize knowledge" (Berrios 1984), "the use or handling of knowledge" (Gregory 1987) or "the overall function" of all "mental abilities" (Glass, Holyoak & Santa 1979), remain scientifically unsatisfying. Further definitions, despite being more elaborated, are either too particular or ambiguous (compare with Costall 1984, Goldberg 1989).

A definition should be as much encompassing as possible in order to appropriately reflect a phenomenon in its distinct manifestations. As a statement, it is a law if it satisfies the following criteria: (a) it is general in the sense that it contains a universal quantifier; (b) it is general in the sense that it does not mention particular individuals, times or places; (c) it has empirical content; (d) it is well-confirmed, and (e) it is well-

entrenched (i.e. it belongs to a theory) (Van der Steen & Kamminga 1991).

The evidence reviewed here allows for an encompassing definition for cognition, from an evolutionary, biophysical and molecular point of view. The definition fulfils the above criteria, and is based on some basic concepts: conservation, evolution, thermodynamics, natural selection and some concepts of the chaos-theory, which are highlighted next.

Conservation and evolution

Conservation is a major attribute of living systems. Here we define this feature as follows:

Throughout a process of change or transformation, conservation is the dynamic matching of innovative features to preexisting ones

It has been suggested elsewhere that replication - a copying process achieved by a special network of interrelationships of components and component-producing processes that produce the same network as the one which has produced them - characterizes the living organization (Maturana & Varela 1980, Csányi & Kampis 1985). This replicative ability (referred to as *autopoiesis*), nevertheless, could not account for evolution of living systems, if only preexisting features were to be maintained. This would be preservation, in the sense of immutability.

Conversely, innovations alone are not capable of inducing evolution if they can not be incorporated to preexisting features (compare with Monod 1985), in such a way that "a small change (...) be accompanied by many simultaneous compensating changes elsewhere" (Marr 1976) or by a "precise readjustment of all other components of the system" (Ashby 1960; see also Macnair 1991).

The ultimate basis of conservation in living organisms lies on the remarkable physico-chemical features of at least three elements: carbon (C), nitrogen (N) and hydrogen (H). The covalent bounds between these elements make it possible reproducible chemical combinations to occur along with the physical properties elicited by the resulting molecular systems. These include the hability to allow for transformations or innovations. These mechanisms sometimes involve the displacement of so extremely small particles that they can be operated analogously to the probability fields of quantum physics, as described by Margenau (1984), but also in accordance to deterministic chaos, as proposed by Crutchfield *et al.* (1986) (compare with Feigenbaum 1978, Prigogine & Stengers 1984, Elskens & Prigogine 1986, Ko 1991). As a result, conservation can be viewed as a "generator of variety" (Simon 1969, Changeux & Dehaene 1989), starting from the molecular level.

Purely mathematical reasoning shows that relatively simple systems can develop quite complex behavior even after slight perturbations, if non-linear interactions are involved (Van der Steen & Kamminga 1991, May 1976). This phenomenon is regarded as important for empirical research since it is exhibited by certain physical and chemical systems. Relatively simple instances of such systems have been long recognized: protobiotic coacerbation of lipoprotein membranes maintained by weak intermolecular attractive forces of the van der Waals type. This system is instructive because it leads us to the considerations about the fundamental relationships between conservation and life at its very beginning.

More complex systems involve associations between peptides and nucleotide sequencies that are capable of self replication, modulation and modification, the most readily recognized example of which is desoxiribonucleic acid synthesis, recombination, transcription and repair. Recently, it has been possible to synthesize very simple self-replicating molecules that are also capable of "successful mutation" (Amato 1992, Hong *et al.* 1992). This evidence constitutes a strong experimental support for the molecular theory of evolution and for the definition of conservation.

However, not only complex molecular (Turing 1952, Britten & Davidson 1969, Strehler *et al.* 1971, Tonegawa 1983), but also microscopic and macroscopic living systems undergo combinations and reciprocal modulation, which are attributable to conservative mechanisms. The successful association of

an *akaryotic* (Forterre 1992) system like a mitochondrion with a *synkaryotic* (*ibidem*) cell, the epigenesis of neural networks (Changeux, Courrège & Danchin 1973) or the phenomenon of ecological successions are instructive examples. In summary, conservation accounts for evolution and for the occurrence of complexity in living systems.

Thermodynamics and conservation

The realization that living organisms obey to the laws of physics can help us understand ourselves, as antecipated by Goethe:

"Wär' nicht das Auge sonnenhaft, die Sonne könnt' es nie erblicken"

Living organisms are open systems. They change mass and energy with the environment. As energy must be conserved throughout transformations (First Law of Thermodynamics, attributed to H. von Helmholtz), an energy inflow to such a system can produce new combinations or alter preexisting ones. This capacity to perform "useful" work has been related to free-energy changes within the system. J.W. Gibbs defined such variations in the free energy function, G, as:

$$\Delta G = H - \Delta T S \quad \text{[Eq. 1]}$$

where H is the enthalpy; T, temperature and S, entropy. In a steady-state, the variation of S (ΔS) equals zero, so that a positive ΔS is balanced by a negative ΔS, and the system feeds on *negentropy* from the surroundings (see Welch 1991). It results that living organisms build up and maintain low-entropic states at the expenses of a net increase in the entropy of the universe.

Nevertheless, free energy tends to increase within a system, i.e., entropy (or chaos) tends to increase internally (Second Law of Thermodynamics). The maintenance of the corresponding changes that can be either vital (e.g. nutrition, photosynthesis) or deleterious (e.g. letal mutations after ionizing radiations) renders the putative conservative forces consistent with the First Law of Thermodynamics.

Similarly, the assumption that the quite complex living systems that we know are just the sustainable portion of the total number of complex living systems that already existed (compare with Stanley 1987) renders conservation compatible with the Second Law of Thermodynamics as well: individual organisms can disappear, by means of extinction or evolution, thanks to their conservative features that maintain not only novel sustainable combinations, but also the highly entropic, unsustainable ones.

Natural selection, environment, and cognition

Whereas conservation agrees well with the first and second laws of thermodynamics, it alone does not suffice as to determine whether particular innovations are to be sustainable or not. It is the dynamic relationships established within the "inner" and "outer" environments of the living system that account for this (compare with Mayr 1963, Jerne 1967, Simon 1969, Monod 1970, Changeux & Danchin 1976, Purves & Lichtman 1980, Lumsden & Wilson 1981, Heidmann, Heidmann & Changeux 1984, Changeux & Dehaene 1989). The role played by these relationships in cutting out the "less fitted" living forms or combinations has been classically described as natural selection in the Darwin-Wallace theory of the evolution of species (Darwin 1859).

Although natural selection is not necessarily intrinsic to individual organisms, living systems can survive novel genetic and epigenetic acquisitions as a result of a somewhat random production of new sustainable patterns for the aforementioned dynamic relationships (compare with Fodor & Pylyshyn 1988), by selection-adaptive forces. This mechanism seems to operate through the matching between entropy-derived combinations and preexisting features (conservation), in an innovative, though coherent manner. A "compensatory", "buffer" or "reset" effect results, by which

conservative mechanisms could no more threaten the sustainability (survival) of the system, at least within a given environmental condition and time.

This feature or "natural probability" is, in fact, the determinant of what has been frequently called cognition. In other words, cognition should be understood as the intrinsic adaptive ability of living systems to environmental changes. As a non-linear function, cognition would operate in molecular, micro- and macroscopic levels, by continuously "resetting" such systems within coherent limits. These limits are made up by the interaction between intrinsic "behavior norms" and multiple environmental factors, eliciting a given number of possible phenotypes. The situation is analogous to the principle of the "attractors" of chaos theory (more detailed descriptions of attractors can be found in Hénon 1976, and Gleick 1987). Summarising our definition:

Cognition is the intrinsic adaption of the system by the unstable search for coherent conservation in respect to the environment

Relationships between intrinsic factors to environmental factors have attracted attention of foremost researchers, and there is work in which "the species-typical environment is regarded as being an equal partner to the genotype" (Morton 1992). French physiologist Claude Bernard (1878) talked about a "perpetual communication of the anatomical element with the medium surrounding it, (...) a continuous relationship of exchange, (...) the continuous mutation of the particles which constitute the living being,(...) [the trait] which ought and could (by itself) suffice to characterize life, (...) the most constant and universal of its manifestations (...)". Young (1964) sustained that "the organism is (or contains) a representation of its environment". Changeux & Dehaene (1989) stated that "the achievement of such adequacy (fitness) with the environment (or with a given cognitive state) would then become the basic criterion for selection".

In the neurophysiology the matching between a *percept* [("in which the correlation of firing among the component neurons is directly determined by the interaction with the outside world and ceases when the stimulation has stopped")] and a *pre-representation* [("analogous to the Darwinian variations")] has been referred to as resonance and its unmatching as dissonance (Changeux 1983, Toulouse, Dehaene & Changeux 1986). In the chaos theory, "the existence and location of attractors is not set in stone but is a function of the environment - though not a linear one" (Firth 1991).

Validation

The above definition of cognition can be tested and validated in a variety of settings. Though it is not the aim of this paper to explore all possibilities, some general exemplification should be of interest.

Ecosystems

Ecosystems are a setting for validation in the macro-scale. In 1838 Grisebach (1814-1879) proposed the modern concept that plants and animals occur in integrated communities. The work of Clements (1874-1945) then consolidated the concept of *biotic unity* of ecosystems, which allows for considerations involving cognition in such systems.

Intrinsic adaptive ability of ecosystems to environmental changes are sometimes referred to as *homeostasis*, a term proposed by Claude Bernard to account for the constancy of the *milieu interne*. A deforestation, for instance, can elicit a dynamic *ecological succession* within a given *biom*. Empirically, this succession operates analogously to the search for coherent conservation. Depending upon the grade of the destruction ("entropy"), it will lead or not to the reconstitution of the ecosystem's original complements and patterns of biodiversity, starting from atypical ones (compare with Gough & Marrs 1990, Overpeck, Bartlein & Webb III 1991).

This dichotomical pattern of non-linear "decision making" has been referred to as analogous to the geometry of fractals and attractors, thus definitely linking cognition with chaos theory

(Feigenbaum 1978, Mandelbrot 1982, Prigogine & Stengers 1984, Elskens & Prigogine 1986, Goldberger, Rigney & West 1990) (Figure 1-a).

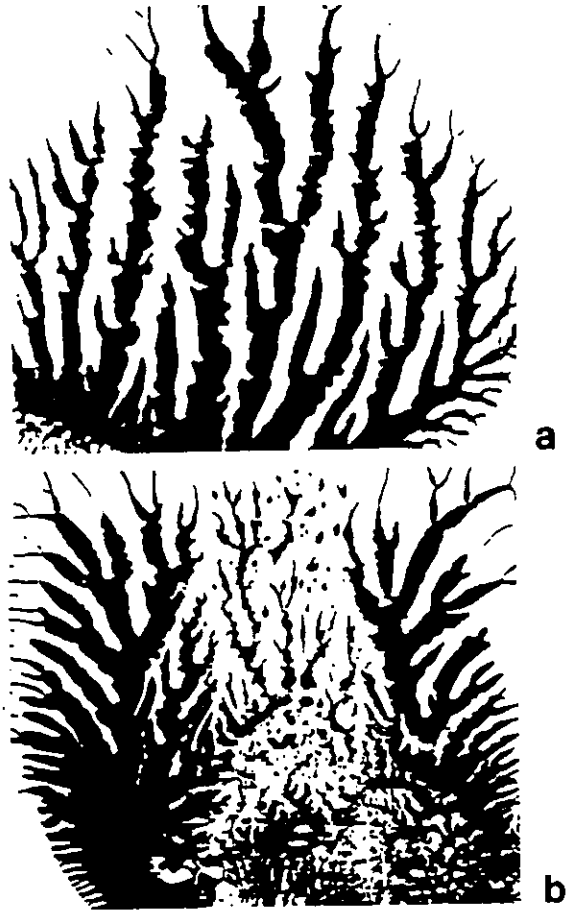


Figure 1. *Fractals can be easily obtained by splitting apart two clean glass slides joined by a drop of ink (a). A break in expected patterns is observed in the center of (b), which corresponds to a track that had been previously covered with a thin layer of fat, resembling the interplay of a "strange attractor".*

Immune systems

A given infectious agent is capable of eliciting a cascade of humoral and cellular events in the synkaryotic host organism, namely the immunological response of the host's immune system. The ability of the immune system in differentially dealing with the "self" and the "non-self" can be regarded as cognitive (compare with Jerne 1967).

At the molecular level, it has been suggested that a bacterial population may switch recognition sequences of its "type I" gene restriction-modification systems by single recombination events and thus being able to maintain an akaryotic analogue of the immune system of variable specificity (Gubler *et al.* 1992).

Alternatively, both somatic recombination and mutation may contribute greatly to an increase in the diversity of antibody synthesized by a single organism (Tonegawa 1983).

Systems of enzymes

Allosteric properties of enzymes have been referred to as "cognitive" (Monod 1970, see also Monod 1966a, Monod 1966b, Schmitt 1967).

Nervous systems

In no other instance, the concept of cognition has been so deeply employed as in the study of the CNS. Therefore, we would like to discuss this setting to some depth.

Nervous systems are syncytium-like neuron and glia systems, and neurons are highly differentiated cells. The process of differentiation is fully impregnated by cognition. It implies that a primitive cell type undergoes a progression of changes that results in changing capacity to perform activities that the cell was previously uncapable of performing (Hayflick 1991).

Neuronal function correlates directly with its structure (Jackson 1932, Chomsky 1979, Changeux 1983b, Babb & Brown 1986, Benes & Bird 1987, Eccles 1990). During development, the neuroarchitecture is achieved by cell proliferation and migration, followed by neurite outgrowth, and finally synaptogenesis (see Lund 1978, Purves & Lichtman 1980, Purves & Lichtman 1985, Chagas, Linden & Lent 1987). In the adult organism, the neuroarchitecture may change adaptively and undergo remodeling, with formation of new synapses and breaking or modification of old ones. All these phenomena should be regarded as cognitive ones, in the inner and outer environmental aspects. As pointed out by Changeux & Dehaene (1989), "the formation of brain architectures is not independent of

cognitive processes but, rather, is deeply impregnated by them, starting from the early stages of postnatal development".

Neuronal plasticity, modulation and transmission have been the major morphofunctional basis for observable cognitive processes in the CNS (compare with Von der Malsburg 1987, Alkon *et al.* 1991). Assessment of cognitive function has been possible through field observations, tests of psychometric intelligence, clinical neuropsychologic tests and laboratory tests (Meyer *et al.* 1988, Colsher & Wallace 1991, Chen 1991). Consequently, the main functions attributed to cognitive processes in the CNS to date are those related to acquisition, storage and retrieval of visuo-spatial, auditory, tactile, kinesthetic, gustatory and olfactory information.

In the human species, the direct assessment of these informations has become possible thanks to our highly developed symbolic representations, such as language and speech. It is noteworthy that language, as a "manner of living", has been "conserved reproductively" in the form of an ontogenetic phenotype (Maturana 1989). Interesting demonstrations of the relationships between genom and language in humans can be found in DeFries *et al.* (1976) and Pennington & Smith (1983).

Mental function

It has been ventured that mental function is based on the ability to keep ideas - units of information, specific representations - distinctly and apart, so that attempts to achieve novel solutions could be made by rearranging the data until the final arrangement corresponds to the desired goal (Kinsbourne 1981).

According to Glassman (1974), creative problem solving consists fundamentally of the activation of global structures derived from past experience and a selection process based on feedback resulting from the interaction of these internal structures with each other or with the environment. The activation of the structures may result from associative (Hebb 1949) and spontaneous arousal processes. Also in this latter instance one can suppose that a basal "noise" corresponding to the oscillation of the system within the fields or boundaries defined by an attractor or set of attractors (generating the rest potential, for instance) can be shifted to an unstable state, influenced by the interplay of another attractor (generating an action potential, in the given instance; compare with Freeman 1988).

The event-related potential (ERP)

The major source of information concerning the neurophysiology of cognitive processes in humans is the scalp-recorded event-related potential (ERP) (John and Schwartz 1978). ERP's can, by habituation and long-term potentiation - an ubiquitous phenomenon of the cerebral cortex (Creutzfeldt 1990, Singer 1990a) - imprint new circuitry patterns in the brain.

Changeux & Dehaene (1989) commented that self-reinforcing circuits may be built at the level of neuronal receptors (Changeux & Heidmann 1987, Crick 1984, Lisman 1985), gene receptors (Britten & Davidson 1969, Monod & Jacob 1961), and at the level of the synapse, on the basis of a sequence of chemical reactions (Changeux, Klarsfeld & Heidmann 1987, see also Petit 1988).

The hypotheses that the CNS produces coherent patterns of linear or cyclic sequences of activity (Grillner 1975, Stent *et al.* 1978, Getting 1981), that neuronal connections are selected according to functional criteria and stabilized if they convey temporally coherent activity (Changeux *et al.* 1973, Dehaene, Changeux & Nadal 1987, Singer 1990b) and that oscillatory responses of neurons can synchronize across spatially separate columns if elicited by stimuli sharing coherent features (Gray *et al.* 1989) have been put forward.

Brains and computational machines

The comparison between information processing in brains and information processing by contemporary computational machines also offers a setting for the validation of our definition of cognition.

Comparative studies have led to "a picture in which disanalogies are more

fundamental than analogies" (Conrad 1989). One of the most significant conclusions derived may be the statement that current computational machines are not capable of cognition just because they are not endowed with the capacity of adaption to environmental changes, in other words, probably because of the lack of coherent conservative processes.

This paradigm is well illustrated by the statement: "brains know what they do, while computers do what they know".

Cognitive impairments

Once cognition relies upon conservative mechanisms, and conservation obeys to the first and second laws of thermodynamics, it follows that cognition is subject to the effects of energy and mass conservation throughout transformations. It is also subject to entropy increase in underlying conservative processes (instances from the medical practice are given by Fukui *et al.* 1991, and Halberg *et al.* 1991).

A cognitive impairment would then take place when conservation overwhelms the system with such an amount of changes that the probability that they could be met to preexisting acquisitions in a sustainable, balanced or coherent manner would decrease critically. When this critical threshold has been achieved, the organism would undergo cognitive dysfunction ("incoherence", "dissonance"), the severity of which correlating to an increase in disease, death or extinction probabilities (compare with some theoretical reasoning by Mézard, Nadal & Toulouse 1986, Nadal *et al.* 1986).

These deviations from stability to irreversibility, in the chaos theory, are possible when there is a "continuous injection of enough energy to drive the system to a high enough excitation that non-linearity becomes appreciable" (Firth 1991).

The encompassing definition of cognition should be of some utility also in the controversial assessment and classification of cognitive impairments, especially in gerontology and pathology. Discrepancy between field and laboratory findings and also weak inter-rater reliability are frequent and contribute to hinder a comprising understanding of these issues (see Avorn 1983). Nevertheless, the assumption that cognition operates through the search for coherent conservation until a given limit, at which entropy reaches a critical point and cognitive impairment supervenes, seems to be a rationale for further studies.

Brain pathology is a quite instructive standpoint. Neuronal populations display varying sensibility to different environmental changes such as hypoxia, vitamin B-deficiency or, much interestingly for our discussion, to the incidence of microscopic aging lesions in the cerebral cortex, e.g., neuritic plaques and neurofibrillary tangles (Ball 1978, Brun & Englund 1981, Braak & Braak 1985, Braak, Braak & Kalus 1989, Dani 1994a). The realization that these aging changes are rather focal dysfunctions affecting susceptible neuronal populations (Tomlinson, Blessed & Roth 1968, Brun & Englund 1981, Gibson 1983, Rafalowska *et al.* 1988) seems to be consistent with the notion of cognitive impairment: they can be regarded as the result of incoherent conservative processes, a dissonant, chaotic deviation ("alienation") of susceptible neurons from expected patterns (Figure 1-b).

Aging

A relatively great amount of evidence of aging at the molecular level has been gathered recently (see, for instance, Strehler *et al.* 1971, Hayflick 1991, Robert 1991, Rose 1991, Macieira-Coelho 1991). Kyriazis (1991) proposed an application of chaos theory for the description of proteolysis, the failure of which could reflect in an increased rate of accumulation of abnormal protein, which has been related to the generation of neuritic plaques and neurofibrillary tangles in the aging brain: when abnormal protein (AP) concentrations rise, there would be an increased synthesis of heat-shock proteins, for example, ubiquitin (Ub). Ubiquitin in turn, would bind covalently to the abnormal protein (compare with Mori, Kondo & Ihara 1987, Harrison & Mullan 1991, Goedert, Spilantini & Crowther 1991) with the help of a number of enzymes. Some of the steps involved would be energy-dependent. Following this, the

polyubiquitinated protein (PUbp) would be degraded by energy-dependent proteases (e.g., unfoldase, isopeptidase, etc.), and the concentration of the abnormal proteins would fall again [It is interesting to remark the similarity of such model with predator-prey models in ecology, "one of the pathbreakers in the study of chaos" (Firth 1991, see also May 1976)]. The initial steps for a mathematical description of this operation would be in the form of the equation:

$$x = G(s)\, f(x) \quad \text{[Eq. 2]},$$

where x is the normalised concentration of the intracellular PUbP, f(x) is the dependence of AP on PUbP in normalised coordinates and G(s) depicts the sequence of intermediate biochemical steps between synthesis of ubiquitin and the onset of the decrease in the concentration of abnormal proteins.

The enhanced incidence of neuritic plaques and neurofibrillary tangles in the human cerebral cortex has been correlated to the severity of cognitive impairment (dementia) as assessed clinically by psychometric tests (Tomlinson *et al.* 1968, Blessed, Tomlinson & Roth 1968, Tomlinson, Blessed & Roth 1970, Wilcock & Esiri 1982, Zubenko *et al.* 1991). Severely demented patients (Alzheimer's disease) usually die within 5-10 years after diagnose of first clinical signs of the disease. It seems that, thanks probably to a feedforward effect (see Rosen 1978), an imbalance between incoherent and coherent conservation operates in the definition of the future cognitive state of the organism; ultimately, in its adaptive ability to environmental changes (compare with Elskens & Prigogine 1986, Freeman 1988, Dani 1994b).

Conclusion

Cognition, defined as the adaption ability of living systems to environmental changes, operates through the unstable search for coherent conservation and is instrumental for life. The term "unstable" refers to dynamic and chaotic features of the search for adaption and accounts also for environmental influences on the destiny of a given cognitive process.

In the words of Firth (1991), "non-linearity is necessary for, and is fundamental to, chaos, but it can also endow stability. Non-linear systems can seek out and maintain essentially the same optimum state in response to a wide variety of external conditions: it is precisely this feature that gives us individuality and continuity".

References

Alkon DL, Amaral DG, Bear MF, Black J, Carew TJ, Cohen NJ, Disterhoft JF, Eichenbaum H, Golski S, Gorman LK, Lynch G, McNaughton BL, Mishkin M, Moyer Jr JR, Olds JL, Olton DS, Otto T, Squire LR, Staubli U, Thompson LT & Wible C (1991) Learning and memory. *Brain Research Reviews*, 16, 193-220.

Amato I (1992) Capturing chemical evolution in a Jar. A system of self-replicating molecules can now spin off a faster-multiplying "mutant" form. *Science*, 255, 800.

Ashby WR (1960) *Design for a Brain*. New York: Wiley.

Avorn J (1983) Biomedical and social determinants of cognitive impairment in the elderly. *Journal of the American Geriatrics Society*, 31, 137-143.

Babb TL & Brown WJ (1986) Neuronal, dendritic, and vascular profiles of human temporal lobe epilepsy correlated with cellular physiology in vivo. *Advances in Neurology*, 44, 949-966.

Ball MJ (1978) Topographic distribution of neurofibrillary tangles and granulovacuolar degeneration in hippocampal cortex of aging and demented patients. A quantitative study. *Acta Neuropathologica* , 42, 73-80.

Benes FM & Bird ED (1987) An analysis of the arrangement of neurons in the cingulate cortex of schizophrenic patients. *Archives of General Psychiatry*, 44, 608-616.

Bernard C (1878) *Leçons sur les Phénomènes de la Vie Communs aux Animaux et aux Végétaux* (Translated by HE Hoff, R Guilleman & L

Guillemin, Springfield, IL: Charles C Thomas Publisher, 1974), as cited by GR Welch (1991)

Berrios GE (1984) Descriptive psychopathology: conceptual and historical aspects. *Psychological Medicine*, 14, 303-313.

Blessed G, Tomlinson BE & Roth M (1968) The association between quantitative measurements of dementia and of senile changes in the cerebral grey matter of elderly subjects. *British Journal of Psychiatry*, 114, 797-811.

Braak H & Braak E (1985) On areas of transition between entorhinal allocortex and temporal isocortex in the human brain. Normal morphology and lamina-specific pathology in Alzheimer's disease. *Acta Neuropathologica*, 68, 325-332

Braak H, Braak E & Kalus P (1989) Alzheimer's disease: areal and laminar pathology in the occipital isocortex. *Acta Neuropathologica*, 77, 494-506.

Britten RJ & Davidson EH (1969) Gene regulation for higher cells: a theory. *Science*, 165, 349-357

Brun A & Englund E (1981) Regional pattern of degeneration in Alzheimer's disease: neuronal loss and histopathological grading. *Histopathology*, 5, 549-564.

Chagas C, Linden R & Lent R (1987) *Developmental Neurobiology of Mammals*. Vatican: Pontificial Academy of Sciences.

Chagas C (1990) Introduçao ao Curso de Pós-Graduaçao em Ciências Biológicas, Area de Concentraçao em Biofísica. "Catálogo das Disciplinas". Rio de Janeiro: Coordenaçao de Pós-Graduaçao do IBCCF, Universidade Federal do Rio de Janeiro.

Changeux JP, Courrège P & Danchin A (1973) A theory of the epigenesis of neural networks by selective stabilization of synapses. *Proceedings of the National Academy of Sciences of the USA*, 70, 2974-2978.

Changeux JP & Danchin A (1976) Selective stabilization of developing synapses as a mechanism for the specification of neuronal networks. *Nature*, 264, 705-712

Changeux JP (1983) *L'Homme Neuronal*. (Paris: Fayard. English translation by L Garey, *Neuronal Man*. New York: Pantheon Books (1985)

Changeux JP, Klarsfeld A & Heidmann T (1987) The acetylcholine receptor and molecular models for short and long term learning. In Changeux JP & Konishi M (Eds.) *The Cellular and Molecular Bases of Learning*. Chichester: Wiley.

Changeux JP & Heidmann T (1987) Allosteric receptors and molecular models of learning. In Edelman G, Gall WE & Cowan WM (Eds.), *Synaptic Function*. New York: Wiley.

Changeux JP & Dehaene S (1989) Neuronal models of cognitive functions. *Cognition*, 33, 63-109.

Chen ACN (1991) Cognitive neuropsychophysiology of thought imagery versus imagination imagery. *International Journal of Neuroscience*, 60, 65-77.

Chomsky N (1979) Le débat entre Jean Piaget et Noam Chomsky. In *Théories du Langage - Théories de l'Apprentissage*. Piattelli-Palmarini M (Ed.) Paris: Le Seuil.

Colsher PL & Wallace RB (1991) Epidemiologic considerations in studies of cognitive function in the elderly: methodology and nondementing acquired dysfunction. *Epidemiologic Reviews*, 13, 1-27.

Conrad M (1989) The brain-machine disanalogy. *Biosystems*, 22, 197-213.

Costall AP (1984) Are theories of perception necessary ? A review of Gibson's the ecological approach to visual perception. *Journal of Experimental Analysis of Behavior*, 41, 109-115

Creutzfeldt OD (1990) Neuronal responses in human and monkey temporal lobe during memory tasks: compartmentalization of memory functions. In *Vision, Memory, and the Temporal Lobe*. New York: Elsevier.

Crick F (1984) Memory and molecular turnover. *Nature*, 312, 101

Crutchfield JP, Farmer JD, Packard NH & Shaw R (1986) Chaos. *Scientific American*, 255, 38-49.

Csányi V & Kampis G (1985) Autogenesis: the evolution of replicative systems. *Journal of Theoretical Biology*, 114, 303-321

Dani SU (1994a) Spatial patterns of plaques and tangles in Alzheimer's disease. Dementia 5:53.

Dani SU (1994b) Larger neuronal size overwhelms maintenance capacity with aging. Medical Hypotheses 42:208-210.

Darwin C (1859) *On the Origin of Species*. London: Murray.

DeFries JC, Vandenberg SG & McClean GE (1976) Genetics of specific cognitive abilities. *Annual Reviews in Genetics*, 10, 179-207.

Dehaene S, Changeux JP & Nadal JP (1987) Neural networks that learn temporal sequences by selection. *Proceedings of the National Academy of Sciences of the USA*, 84, 2727-2731.

Eccles Sir J (1990) A unitary hypothesis of mind-brain interaction in the cerebral cortex. *Proceedings of the Royal Society of London. B* 240, 433-451.

Elskens Y & Prigogine I (1986) From instability to irreversibility. *Proceedings of the National Academy of Sciences of the USA*, 83, 5756.

Feigenbaum MJ (1978) Quantitative universality for a class of nonlinear transformations. *Journal of Statistical Physics*, 19, 25 -52.

Firth WJ (1991) Chaos - predicting the unpredictable. *British Medical Journal*, 303, 1565-1568.

Fodor J & Pylyshyn Z (1988) Connectionism and cognitive architecture: a critical analysis. *Cognition* 28, 3-71 .

Forterre P (1992) Neutral Terms. *Nature*, 355, 305.

Freeman WJ (1988) Strange attractors that govern mammalian brain dynamics shown by trajectories of electroencephalographic (EEG) potential. *Transactions on Circuits and Systems*, 35, 781.

Fukui Y, Hoshino K, Hayasaka I, Inouye M & Kameyama Y (1991) Developmental disturbance of rat cerebral cortex following prenatal low-dose gamma-irradiation: a quantitative study. *Experimental Neurology*, 112, 292-298.

Getting PA (1981) Mechanism of pattern generation underlying swimming in *Tritonia*. I. Neuronal network formed by monosynaptic connections. *Journal of Neurophysiology*, 46, 65-79.

Gibson PH (1983) Form and distribution of senile plaques seen in silver-impregnated sections in the brains of intellectually normal elderly people and people with Alzheimer-type dementia. *Neuropathology and Applied Neurobiology*, 9, 379-389.

Glass AL, Holyoak KJ & Santa JL (1979) *Cognition*. Reading, MA: Addison-Wesley.

Glassman RB (1974) Selection processes in living systems: role in cognitive construction and recovery from brain damage. *Behavioral Science*, 19, 149-165.

Gleick J (1987) *Chaos: Making a New Science*. New York: Viking.

Goedert M, Spilantini MG, & Crowther RA (1991) Tau proteins and neurofibrillary degeneration. *Brain Pathology*, 1, 279-286.

Goldberg E (1989) Gradiental approach to neocortical functional organization. *Journal of Clinical and Experimental Neurophysiology*, 11, 489-517.

Goldberger AL, Rigney DR & West BJ (1990) Chaos and fractals in human physiology. *Scientific American*, 262, 35-41.

Gough MW & Marrs RH (1990) A comparison of soil fertility between semi-natural and agricultural plant communities: implications for the creation of species-rich grassland on abandoned agricultural land. *Biological Conservation*, 51, 83-96

Gray CM, König P, Engel AK & Singer W (1989) Oscillatory responses in cat visual cortex exhibit inter-columnar synchronization which reflects global stimulus properties. *Nature*, 338, 334-337.

Gregory RL (1987) *The Oxford Companion to the Mind*. Oxford, England: Oxford University Press.

Grillner S (1975) Locomotion in vertebrates. Central mechanisms and reflex interaction. *Physiological Review*, 55, 247-304.

Gubler M, Braguglia D, Meyer J, Piekarowicz A & Bickle T (1992) Recombination of constant and variable modules alters DNA sequence recognition by type IC restriction-modification enzymes. *The EMBO Journal*, 11, 233-240.

Halberg FE, Kramer JH, Moore IM, Wara WM, Matthay KK & Ablin AR (1991) Prophylatic cranial irradiation dose effects on late cognitive function in children treated for acute lymphoblastic leukemia. *International Journal of Radiation Oncology in Biology and Physics*, 22, 13-16.

Harrison PJ & Mullan MJ (1991) Alzheimer's disease: the significance of a ß-amyloid precursor protein gene mutation. *Current Opinion in Neurology and Neurosurgery*, 4, 908-913.

Hayflick L (1991) Aging under glass. *Mutation Research*, 256, 79-80.

Hebb DO (1949) *Organization of Behavior*. New York: Wiley.

Heidmann A, Heidmann T & Changeux JP (1984) Stabilisation sélective de représentations neuronales par résonance entre "pré-représentations" spontanées du réseau cérébral et "percepts" évoqués par interaction avec le monde extérieur. *Comptes Rendues Académie de Sciences, Paris (série 3)*, 229, 839-844.

Hénon M (1976) A two-dimensional mapping with a strange attractor. *Communications in Mathematical Physics*, 130, 69-77.

Hong J-I, Feng Q, Rotello V & Rebek Jr. J (1992) Competition, cooperation, and mutation: improving a synthetic replicator by light irradiation. *Science*, 255, 848.

Jackson H (1932) In Taylor J (Ed.). *Selected Papers (Vol.2)*. London: Hodder & Stoughton.

Jerne N (1967) Antibodies and learning: selection versus instruction. In *The Neurosciences*. Quarton G., Melnechuck T & Schmitt FO (Eds.). New York: Rockfeller University Press.

John ER & Schwartz EL (1978) The neurophysiology of information processing and cognition. *Annual Reviews in Psychology*, 29, 1-29.

Kinsbourne M (1981) Cognitive deficit and the unity of brain organization: Goldstein's perspective updated. *Journal of Communication Disorders*, 14, 181-194 .

Ko MSH (1991) A stochastic model for gene induction. *Journal of Theoretical Biology*, 153, 181-194.

Kyriazis M (1991) Applications of chaos theory to the molecular biology of aging. *Experimental Gerontology*, 26, 569-572.

Lisman JE (1985) A mechanism for memory storage insensitive to molecular turnover: a bistable autophosphorylating kinase. *Proceedings of the National Academy of Sciences of the USA*, 82, 3055-3057.

Lumsden C & Wilson EO (1981) *Genes, Mind and Culture: the Coevolutionary Process*. Cambridge, MA: Harvard University Press.

Lund RD (1978) *Development and Plasticity of the Brain*. New York: Oxford University Press.

Macieira-Coelho A (1991) Dérégulation du flux d'information dans l'organisme âgé. *Comptes Rendues de Société de Biologie*, 185, 110-120 .

Macnair MR (1991) Why the evolution of resistance to anthropogenic toxins normally involves major gene changes: the limits to natural selection. *Genetica* 84, 213-219 .

Mandelbrot BB (1982) *The Fractal Geometry of Nature*. New York: Freeman.

Margenau H (1984) *The Miracle of Existence*. Woodbridge, Connecticut: Ox Bow Press.; cited by Eccles, Sir J. (1990).

Marr D (1976) *Philosophical Transactions of the Royal Society of London, B*, 275, 483-519, quoted in Shallice, 1981.

Maturana H (1989) Lenguaje y realidad: el origen de lo humano. *Archivos de Biologia y Medicina Experimentales*, 22, 77-81.

Maturana HR & Varela FJ (1980) *Autopoiesis and Cognition, Boston Studies in the Philosophy of Sciences*, 42. Boston: D.Reidel.

May RM (1976) Simple mathematical models with very complicated dynamics. *Nature*, 261,459-67.

Mayr E (1963) *Animal Species and Evolution*. Cambridge, MA: Harvard University Press.

Mézard M, Nadal JP & Toulouse G (1986) Solvable models of working memories. *Journal de Physique (Paris)* 47, 1457-1462.

Meyer DE, Osman AM, Irwin DE & Yantis S (1988) Modern mental chronometry. *Biological Psychology*, 26, 3-67.

Monod J (1966a) From enzymatic adaption to allosteric transitions. *Science*, 154, 475-483.

Monod J (1966b) On the mechanism of molecular interactions in the control of cellular metabolism. *The Upjohn Lecture of The Endocrine Society USA*, 78, 412-425.

Monod J (1970) *Le Hasard et la Nécessité*. Paris: Le Seuil.

Monod J & Jacob F (1961) General conclusions: teleonomic mechanisms in cellular metabolism, growth and differentiation. *Cold Spring Harbor Symposium for Quantitative Biology*, 26, 389-401.

Monod JL (1974) Sobre a Teoria Molecular da Evolução. In Rom Harré (Ed.), *Problems of Scientifical Revolution. Progress and Obstacles to Progress in the Sciences* (Portuguese translation by L Hegenberg & OS da Mota). Belo Horizonte: Itatiaia.

Mori H, Kondo J & Ihara Y (1987) Ubiquitin is a component of paired helical filaments in Alzheimer's disease. *Science*, 235, 1641-1644.

Morton J (1992) Between brain and culture. Origins of the Modern Mind: Three Stages in the Evolution of Culture and Cognition, by Donald M, Harvard University Press:1991. *Nature*, 355, 31.

Nadal JP, Toulouse G, Changeux JP & Dehaene S (1986) Networks of formal neurons and memory palimpsests. *Europhysics Letters* 1, 532-542.

Overpeck JT, Bartlein PJ & Webb III T (1991) Potential magnitude of future vegetation change in eastern North America: comparisons with the past. *Science*, 254, 692-695.

Pennington BE & Smith SD (1983) Genetic influences on learning disabilities and speech and language disorders. *Child Development*, 54, 369-387.

Petit TL (1988) The neurobiology of learning and memory: elucidation of the mechanisms of cognitive dysfunction. *Neurotoxicology*, 9,413-428.

Prigogine I & Stengers I (1984) *Order out of Chaos: Man's New Dialogue with Nature*. Toronto:Bantam.

Purves D & Lichtman JW (1980) Elimination of synapses in the developing nervous system. *Science*, 210, 153-157.

Purves D & Lichtman JW (1985) *Principles of Neural Development*. Sunderland, Sinauer.

Rafalowska J, Barcikowska M, Wen GY & Wisniewsky HM (1988) Laminar distribution of neuritic plaques in normal aging, Alzheimer's disease and Down's syndrome. *Acta Neuropathologica*, 77, 21-25.

Robert L (1991) Mécanismes cellulaires et moléculaires du vieillessement: une revue. *Comptes Rendues de Société de Biologie*, 185, 97-109.

Rose MR (1991) *Evolutionary Biology of Aging* New York: Oxford University Press.

Rosen R (1978) Feedforwards and global system failure: a general mechanism for senescence. *Journal of Theoretical Biology*, 74, 579-590.

Schmitt FO (1967) Makromolekulare Datenverarbeitung im Zentralnervensystem. *Klinische Wochenschrift*, 45, 863-868 .

Shallice T (1981) Neurological impairment of cognitive processes. *British Medical Bulletin* 37, 187-192.

Simon HA (1969) *The Sciences of the Artificial* Cambridge, MA: MIT Press.

Singer W (1990a) Ontogenetic self-organization and learning. In Mc Gaugh JL, Weinberger NM & Lynch G (Eds.) *Brain Organization and Memory: Cells, Systems and Circuits*. New York: Oxford University Press.

Singer W (1990b) Search for coherence: a basic principle of cortical self-organization. *Concepts in Neuroscience*, 1, 1-26 .

Stanley S (1987) *Extinction*. New York: Freeman.

Stent GS, Kristan WB, Friesen WO, Ort CA, Poon M & Calabrese RL (1978) Neuronal generation of the leech swimming movement. *Science*, 200, 1348-1356.

Strehler B, Hirsch G, Gusseck D, Johnson R & Bick M (1971) Codon-restriction theory of aging and development. *Journal of Theoretical Biology*, 33, 429-474.

Tomlinson BE, Blessed S & Roth M (1968) Observations on the brains of non-demented old people. *Journal of the Neurological Sciences*, 7, 331-356.

Tomlinson BE, Blessed S & Roth M (1970) Observations on the brains of demented old people. *Journal of the Neurological Sciences*, 11, 205-242.

Tonegawa S (1983) Somatic generation of antibody diversity. *Nature*, 302, 575-581.

Toulouse G, Dehaene S & Changeux JP (1986) Spin glass model of learning by selection. *Proceedings of the National Academy of Sciences of the USA*, 83, 1695-1698.

Turing AM (1952) The chemical basis of morphogenesis. *Philosophycal Transactions of the Royal Society of London*, 237, 37-72

Van der Steen WJ & Kamminga H (1991) Laws and natural history in biology. *British Journal of the Philosophy of Science*, 42, 445-467 .

Von der Malsburg C (1987) Synaptic plasticity as basis of brain organization. In JP Changeux & M Konishi (Eds.) *The Neural and Molecular Bases of Learning*. Chichester: Wiley.

Welch GR (1991) Thermodynamics and living systems: problems and paradigms. *Journal of Nutrition*, 121, 1902-1906 .

Wilcock GK & Esiri MM (1982) Plaques, tangles and dementia. A quantitative study. *Journal of the Neurological Sciences*, 56, 343-356.

Young JZ (1964) *A Model of the Brain*. Oxford: Claredon Press.

Zubenko GS, Moossy J, Martinez J, Rao G, Claassen D, Rosen J & Kopp U (1991) Neuropathologic and neurochemical correlates of psychosis in primary dementia. *Archives of Neurology*, 48, 619-624.

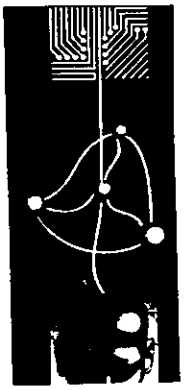
Acknowledgements

Work supported by CAPES (Brazil). Eugênia Vale Marques, M.Sc., kindly collaborated with one reference. Valuable comments on the manuscript by Professor JEH Pittella are kindly acknowledged.

Correspondence should be sent to:

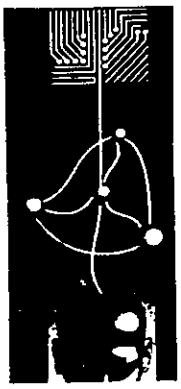
Sérgio U. Dani, MD
Institut für Neuropathologie der MHH
Konstanty-Gutschow-Str. 8
30625 Hannover
Germany
Fax: (0049-511) 532 4550
E-mail: dani@dhvmhh1.bitnet
or dani@mhrz.mh-hannover.de

Anotações



MISCELÂNEA I

Anotações



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

AQUISIÇÃO DE CONHECIMENTO DE TEXTOS UTILIZANDO TÉCNICA CONEXIONISTA

I. R. Guilherme
DEMAC/IGCE/UNESP-Rio Claro
A. F. Rocha
DFB/IB/UNICAMP

RESUMO

O sistema JARGÃO é uma ferramenta para aquisição de conhecimento de textos em liguagem natural. O conhecimento obtido é estruturado em redes neurais simbolicas hierárquicas: redes de conceitos, redes de classes e redes de teorias. O conhecimento obtido pode ser utilizado para diversas finalidades.

1. INTRODUÇÃO

A linguagem falada e escrita tem um importante papel no desenvolvimento do aprendizado e na comunicação das pessoas. Com o surgimento dos computadores e fazendo uso do desenvolvimento dos conhecimentos relacionados com a linguagem, tem se desenvolvido a área do processamento da linguagem natural (PLN), que trabalha com problemas como: interfaces em linguagem natural para sistemas computacionais, interpretação e geração de textos, tradução automática, recuperação e filtragem de informação, etc. O desenvolvimento desta área tem sido feito por pesquisadores de duas linhas distintas em suas concepções básicas:

- os cognitivistas: assumem que a unidade básica de processamento é o símbolo e que todas as regras para o processamento são definidas previamente. São fortemente influenciados pelos trabalhos do Chomsky ([CHOM57],[CHOM65]);
- os conexionistas: assumem que a unidade básica de processamento é o neurônio e que estes são agregados (conexões) para a construção de estruturas mais complexas (relações, regras). As relações são obtidas a partir de um conjunto de exemplos ([McCL86][RUME86]).

A aplicação de técnicas conexionista no PLN tem alcançado resultados expressivos em problemas denominados de baixo nível, como reconhecimento de caracteres, reconhecimento e produção da fala, modelagem do efeito do contexto na compreensão da linguagem natural, etc. Para tratar problemas de linguagem de mais alto nível a técnica não tem produzido resultados tão expressivos. Isto ocorre porque o processamento de alto nível é predominantemente simbólico. Porém sabe-se que este processamento simbólico é feito por redes de neurônios no cérebro.

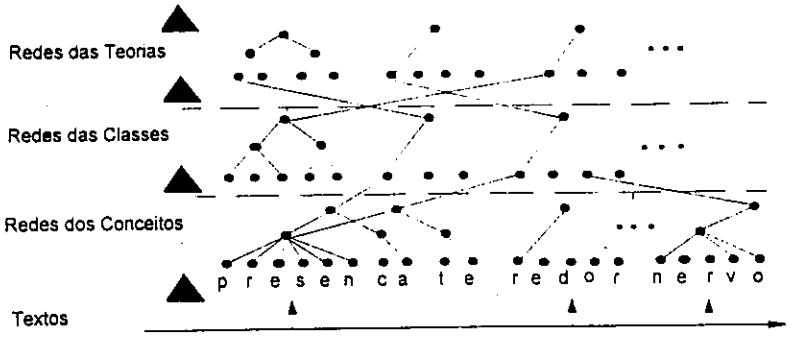


Figura 1: Estrutura Hierárquica das Redes no Jargão

A solução portanto consiste em definir um modelo de neurônio que faça processamento numérico e simbólico. Este trabalho utiliza a definição de neurônio ([ROCH92]) que possibilita o processamento numérico e simbólico na análise de textos.

A análise de entrevista, relatório ou legislação é uma atividade complexa e trata-se de um requisito comum em vários setores da atividade humana. Análise automatizada de textos é também uma tarefa complexa e as implementações existentes consistem em produzir representação detalhada de um número pequeno de textos. Com a evolução da capacidade de armazenamento e processamento dos sistemas computacionais aumentou-se a necessidade de sistemas para a análise de grande número de textos ([JACO93]). Este tipo de análise, devido a sua complexidade computacional não pode ser feita de forma rigorosa. A forma de análise implementada no sistema JARGÃO e que será descrita neste trabalho, consiste em fazê-la da mesma forma que nós humanos trabalhamos, ou seja explorando a imprecisão e a incerteza, e fazendo uso de outra importante heurística que é o contexto.

2. ESTRUTURA DO SISTEMA JARGÃO

O sistema Jargão gera três níveis de redes hierarquicamente organizadas: redes de conceitos, redes das classes e as redes das teorias. Estas redes são criadas contendo as informações comuns de um grupo de textos, contidos na base de dados em linguagem natural analisada. A primeira rede consiste da rede de conceitos que é construida com as palavras ou associações de palavras que representam os conceitos mais significativos e freqüentes presentes nos textos. A redes de classes é construída com as classes mais significativas e freqüentes nos textos. e que consiste da conexão da saída das redes de conceitos nas entradas das redes de classes. Essas redes de classes são usadas como entradas das redes de teorias. que consistem dos textos padrões ou significativos contidos na base de dados (Figura 1).

3. OBTENÇÃO DAS REDES DE CONCEITOS

Um conceito pode ser um conceito simples ou composto. O conceito simples consiste de ter associado apenas uma palavra. Conceitos compostos podem ter necessidade de várias palavras. Neste contexto, a rede de conceitos simples é construída como redes neurais de três camadas. Aos neurônios da camada

inferior, denominado de entradas serão associado códigos ASCII. Os neurônios da camada intermediária são de agregação, e os neurônios das camadas superior são denominados de saída. Os neurônios da camada inferior podem ser agrupados em partes. A parte denominada Germe é composta pelos neurônios associados aos caracteres iniciais que são idênticos nas palavras incorporadas na mesma rede e que são agregados por um neurônio da camada intermediária. O Germe serve como índice dos conceitos. A parte denominada Halo é composta pelos neurônios associados aos caracteres que complementam o germe na formação da palavra e que são agregados por um neurônio da camada intermediária. A rede gerada para conceitos diferentes das existentes não necessitam de camadas intermediárias e os neurônios de entradas são agregados por um neurônio de saída. Nas outras redes, a agregação do germe com os halos é feita por neurônios de saída.

As redes de conceitos compostos são construídas utilizando redes de conceitos simples já criadas. A obtenção dos conceitos compostos pode ser feita utilizando os recursos do sistema ou pelo próprio usuário, que neste caso consiste em agrupar as redes de conceitos simples na formalização do conceito composto.

Todas as palavras contidas nos textos são lidas e os módulos das redes de conceitos são geradas de acordo com a topologia descrita acima.

O usuário faz a análise dos conceitos encontrados e aqueles que não têm relação com o contexto trabalhado ou que sejam muito pouco freqüentes são eliminados. Deve-se também agrupar os conceitos que são sinônimo em um único modulo de rede. Gera-se então o dicionário de conceitos, sobre o qual passa-se a operar.

4. OBTENÇÃO DAS REDES DE CLASSES

4.1 Definição da sintaxe

O primeiro aspecto a ser levando em consideração para a geração das classes é verficar a característica da base analisada no que refere-se o tipo de expressões nela existente. As expressões podem ser: descritivas (contém informações ou a descrição de um símbolo) ou procedural (descreve uma ação, e a ação é representada por verbos).

Portanto, de acordo com a característica das expressões contidas nos textos analisados, deve-se definir inicialmente quais os conceitos que são termos chaves (símbolos ou verbos), e as outros conceitos devem ser classificadas como complemento. Podem existir conceitos classificados como termo chave e complemento.

O passo seguinte consiste em definir as classes, e que pode ser feita utilizando as classes gramaticais encontradas na gramática transformacional, gramática de casos, etc. Pode-se, por exemplo, adotar, utilizando as definições de gramática de casos as seguintes classes: transitividade do verbo (transitivo direto, transitivo indireto, bitransitivo), agente, coagente, lugar, objeto, origem e destino, transporte, etc.

Utiliza-se conceitos conhecidos em neurofisiologia que é a noção de transmissor, receptor e controlador para codificar as informações das classes lingüísticas nos neurônios. A afinidade transmissor/receptor é utilizado para codificar as regras sintáticas que especificam as classes de concatenação nesta sintaxe. Neste contexto deve-se definir:

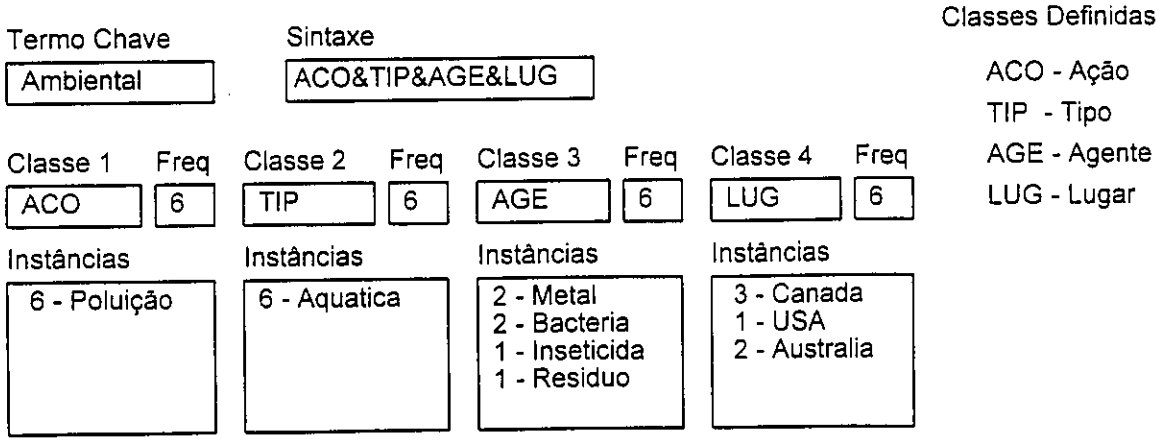


Figura 2: Apresentação das Instâncias de uma rede de classe.

a) os trasmissores que são representado com símbolo como vtd, age, coa, lug, obj, ori, etc, codificando respectivamente agente, coagente, lugar, objeto, origem e destino, etc.

b) os receptores que são representados como , VTD, VTI, AGE, COA, LUG, OBJ, ORI, etc.

c) os controladores do tipo vtdAGE, ageCOA, vtdOBJ com objetivo de implementar regras sintáticas condicionais, como por exemplo:

AGE ^ ageOBJ >> OBJ ou
VTD ^ vtdAGE >> AGE

Utilizando-se as regras acima, define-se os símbolos associados aos transmissores, receptores e moduladores que representam a sintaxe que será adotada. Tendo sido feito a definição ou a escolha da sintaxe a ser utilizada o usuário deverá:

a) atribuir diferentes classes de receptores para os termos do dicionário. classificados como termo chave, usando as cadeias correspondentes (AGE, VTD, VTI, etc).

b) atribuir diferentes classes de transmissores ou moduladores para os termos do dicionário, classificados como complementos, usando as cadeias correspondentes (age, obj, lug, etc ou vtdAGE, ageCOA, etc).

Tendo sido feito as definições para os conceitos do dicionário será então gerado as redes de classes, usando como conjunto de treinamento as frases dos textos.

4.2 Obtenção das classes

As redes de classes são estruturas de mais alto nível que são instanciadas pelos conceitos. A topologia consite em: serem criadas tantas redes de classes quanto forem o número de termos chaves definidos; o primeiro neurônio na rede de classes é o termo chave, esse neurônio caracteriza o módulo; os outros neurônios de entrada são classes complementos e eles produzem receptores para diferentes categorias sintáticas aceita pelos verbos e seus próprios complementos. Nesse caso, são criadas tantos neurônios de entradas quantos forem as classes sintáticas requeridas pelos termos chaves e seus complementos.

A definição sintatica funciona como heurística na busca das combinações. Esta codificação especifica as informações contextuais desejadas pelo usuário. Quanto mais restrita for a sintaxe definida maior será seu efeito sobre a explosão combinatorial.

Utilizando as definições sintáticas podemos criar classes como estruturas de representação do conhecimento conhecidas (frames, dependência conceitual, redes semânticas).

4.3 Definição semântica das classes

Cada uma das redes de classe associadas aos termos chaves é mostrada (Figura 2) para que o usuário elimine as classes incoerentes com o contexto e defina a semântica das classes corretamente construídas. Para auxiliar nesta definição, são mostradas as frases que foram utilizadas no treinamento e as estruturas sintáticas de todas as classes obtidas. Tendo estas informações o usuário deverá:

a) guardar as classes que inequivocadamente definem um significado específico no contexto trabalhado. Poderá associar uma frase ou um símbolo que represente aquela classe, caso contrário, o sistema cria um símbolo para representar a classe.

b) descartar as classes que sejam semanticamente e sintaticamente incoerentes.

Nas situações onde os resultados obtidos não são satisfatórios deve-se refazer a definição sintática afim de corrigir as eventuais distorções.

5. OBTENÇÃO DAS REDES DAS TEORIAS

Este nível é semelhante ao nível das redes de classes. Inicialmente constroi-se um dicionário contendo as classes mais freqüentes. Define-se uma sintáxe que represente as teorias desejadas, e então associa-se as classes. Em função das definições das classes e das suas ocorrências nos textos são gerados as redes de teorias. Em [ROCH92b] descreve-se a geração de teorias utilizado como classe os conceitos linguísticos de tema e rema. As redes de teorias encontradas são mostradas aos usuários.

6. APLICAÇÕES

O sistema JARGÃO tem sido utilizado para aquisição de conhecimento de um conjunto de textos em linguagem natural, numa dada área de especialização. O conhecimento obtido tem sido utilizado no desenvolvimento de sistemas especialistas conexionistas([ROCH92b et all]), para a estruturação dos casos em sistemas de raciocínio baseados em casos, etc.

Com o crescimento do volume de informação disponiveis tem tornado-se necessário a criação de eficientes sistemas de indexação de textos, filtragem e recuperação de informações, obtenção de padrões nos textos, etc. Alguns dos sistemas desenvolvidos ([JACO93]) tem criado mecanismos para o usuário codificar o seu conhecimento que é utilizado nas consultas. O conhecimento (crenças, metas, etc) pode ser constantemente ajustado. O sistema JARGÃO pode operar desta forma, ou ser utilizado para obter o conhecimento a ser utilizado em consultas e ao também para recuperar os textos desejados. A eficiência da recuperação da informação depende do nivel de conhecimento especificado. Caso o usuário especifique o conhecimento a nível de conceito a quantidade de informações recuperadas será enorme e conterá informações indesejadas, se especificar a nível de classes a eficiência deverá aumentar considerávelmente e a nível de teoria obtem-se somente as informações associadas com a teoria que se desejada.

7. BIBLIOGRAFIA

[CHOM57] Noam Chomsky. Sintactic Structures, the Hague: Mouton(1957).

[CHOM65] Noam Chomsky. Aspects of the theory of sintax, MIT Press, Cambridge, USA(1965)

[JACO93] P. S. Jacobs, L. F. Rau. Inovations in text interpretation, Artificial Intelligence, Vol 63, 1993..

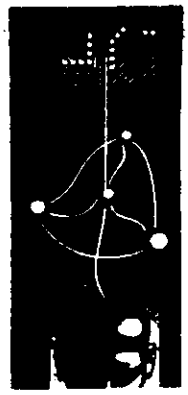
[McCL86] J. L. McClelland, D. E. Rumelhart, Parallel Distributed processing: Explorations in Microstructure of Cognition (Vol. 2), Cambridge, MA: Bradford Books, 1986.

[ROCH92a et all] A. F. Rocha, I.. R. Guilherme, M. Theoto, A. M. K. Miyadahira, and M. S. Koizumi, A neural Network for extracting Knowledge from Natural Language Data Bases, IEEE Transactions on Neural Network, Vol. 3, Num. 5, September 1992.

[ROCH92b et all] A. F. Rocha, I. R. Guilherme, R. J. Machado, Knowledge Aquisition: An Connectionist Approach, Proceedings of 3th Annual Simposiun of the International Association of Knowledge Engennering- IAKE, November, 1992, Washington, USA

[ROCH92c] A. F. Rocha, *The theory of Brains and Machines*, in Lectures and Notes in Artificial Intelligence, New York: Springer Verlag, 1992, 400pp.

[RUME86] D. E. Rumelhart, J. L. McClelland, Parallel Distributed processing: Explorations in Microstructure of Cognition (Vol. 1), Cambridge, MA: Bradford Books,1986.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá. 24 a 27 de outubro de 1994

Implementação de um Jogo de Damas Utilizando uma Rede Neural Multi-Camadas Associada a uma Heurística.

Paulo André S. Perez
USP - ICMSC -SCE
13560-150 - São Carlos - SP - Brasil
e-mail: perez@icmsc.usp.br

Roseli Ap. Francelin *
USP - ICMSC - SCE
13560-150 - São Carlos - SP -Brasil
e-mail: rafrance@icmsc.usp.br

Maria Carolina Monard *
USP - ICMSC - SCE
13560-150 - São Carlos - SP - Brasil
e-mail: monard@.icmsc.usp.br

Sumário

Neste trabalho discutimos a união da Programação Simbólica à Conexionista na implementação de um Jogo de Damas. Uma heuristica é proposta de forma a encontrar o melhor movimento a ser efetuado pelo computador, isto em conjunto com uma Rede Neural Multi-Camadas que é treinada e utilizada para estimar quais serão os movimentos do usuário. O modelo é analisado, tendo como foco as estruturas de dados, a heurística adotada, a Rede Neural, o processo de aprendizagem e o algoritmo que gera, com base na saída da Rede, os vários possíveis movimentos do usuário.

Uma discussão sobre a capacidade de aprendizagem do Jogo, é feita a partir dos resultados obtidos. Sugestões para futuros trabalhos são feitas.

* Trabalho desenvolvido com o apoio da FAPESP

I - INTRODUÇÃO

Tipicamente vemos o conflito entre a os adeptos da Inteligência Artificial Simbólica e os da Conexionista. Contudo é fato que ambas buscam a representação do conhecimento e o processo de aprendizagem. cada uma apresentando pontos fortes e fracos em determinadas aplicações. Nada mais natural então. do que propor a união destas de forma a resolver um dado problema. Como exemplo. dada a facilidade de implementação. enfocaremos um Jogo de Damas. onde teremos um heurística que encontrará quais são os possíveis movimentos de um jogador. e dentre eles escolherá o melhor com base no contra-movimento a ser efetuado pelo usuário. o qual será estimado por uma Rede Neural Multi-Camadas.

Este trabalho. é organizado com segue. Na Seção II, propomos a Rede Neural Multi-Camadas e discutimos sua utilização. Na Seção III. mostramos o algoritmo da heurística adotada. Na Seção IV. é abordada a implementação. Finalmente. conclusões e futuros trabalhos são apresentados na Seção V.

II - A REDE NEURAL

Redes Neurais Artificiais (RNA) possuem uma grande diversidade de aplicações. dentre elas: processamento de sinais [1]. reconhecimento de padrões [1-2] e otimização [3-5]. Dentre as RNA. destacamos a chamada Rede Neural Multi-Camadas (Figura 1). isto dada a sua capacidade de aprendizagem e generalização.

Utilizaremos uma RNA Multi-Camadas para representar o conhecimento do usuário no Jogo de Damas. de tal forma que as entrada da Rede correspondem a atual disposição das pedras no tabuleiro. e a saída. o movimento que o usuário faria diante desta situação (Figura 2).

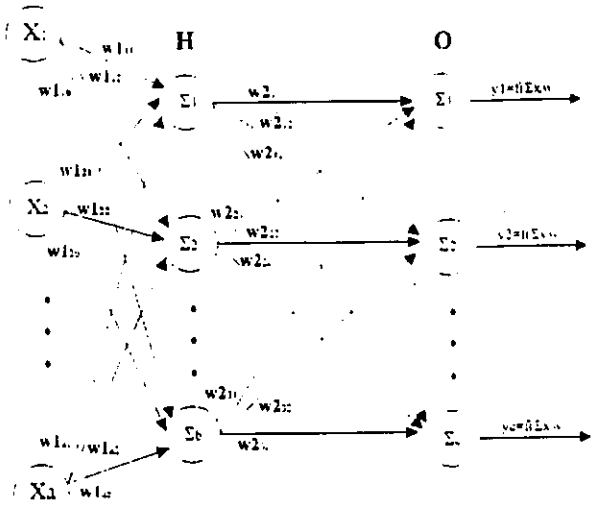


Figura 1 - Rede Neural Multi-Camadas

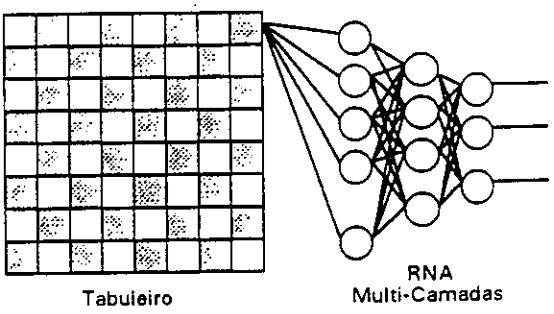


Figura 2 - Modelo Adotado

Foi adotada uma matriz 8x8 de números inteiros. doravante denota por T. para representar o tabuleiro. onde um elemento pode assumir um dos seguintes valores: 0 para posição vazia; 1 para posição ocupada por uma peça do jogador número 1; 2 para peça do jogador número 2; -1 para representar uma *dama* do jogador número 1; e -2 para uma *dama* do jogador número 2.

A matriz T corresponde à entrada da Rede. ou seja. todos os elementos da matriz estão ligados a todos os nós da primeira camada oculta da Rede e vice-versa.

O movimento do usuário é composto por dois pares de coordenadas. que correspondem à linha/coluna inicial e final da peça a ser movida. Como as saídas da Rede. que no nosso modelo utiliza a função sigmóide. estão entre 0 e 1. é necessário que codifiquemos estas coordenadas em seus equivalentes binários (000 a 111). Logo teremos com saída um vetor M de 12 elementos

Desta forma fica estabelecido que temos uma Rede com 64 entradas. 2 camadas ocultas e

12 nós na camada de saída. As quantidades de nós nas camadas ocultas foram estabelecidas com base em testes, adotando-se então 48 nós na primeira camada oculta e 24 na segunda.

Basicamente o que desejamos da Rede é tentar estimar os movimentos do usuário no jogo, ou seja, dada uma matriz **T**, obter como saída o movimento(s) que o usuário fará. Para conseguir isto, devemos treinar a Rede com o maior número possível de pares **T,M** , que são obtidos ao longo de partidas anteriores entre o computador e o usuário em questão. Assim, teremos não só uma "memória" das partidas anteriores, como também uma previsão de possíveis movimentos, isto graças à generalização oferecida pela Rede.

Contudo existe um problema, neste caso específico, que é o fato de que uma RNA Multi-Camadas, num dado instante, dada uma única entrada, matriz **T**, temos somente uma única saída, vetor **M**. Como num jogo, existem diversos movimentos possíveis, e também objetivando explorar ao máximo a capacidade de abstração da Rede, definimos uma função que com base na saída gerada pela Rede, obtém todos os possíveis movimentos derivados da mesma, de acordo com o que se segue:

- Para cada nó de saída temos que o respectivo bit assumirá o valor:

- 0 , se a saída for menor ou igual a 0.1;
- 1 , se a saída for maior ou igual a 0.9;
- 0 e depois 1 , se a saída estiver entre 0.1 e 0.9, ou seja, se tivermos uma saída neste intervalo, primeiro geraremos um movimento assumindo que este bit é 0 e depois outro assumindo que o mesmo bit é 1.

Definimos ainda, um grau de certeza, ou razão de certeza (*ratio*) que corresponderá à confiança que temos deste movimento ser provável, e que é dada por:

$$Ratio = \prod_{i=1}^{12} p_i$$

onde

$p_i = (1 - M_i)$, se for assumido que o bit i é 0;
$p_i = M_i$, se for assumido que o bit i é 1.

Este *ratio* vem do fato de acreditarmos que movimentos já aprendidos pela Rede, e que portanto com pequeno erro, provavelmente ocorrerão novamente, ao passo que situações "desconhecidas" gerarão movimentos "estimados", com valores entre 0.1 e 0.9 , e que portanto são menos prováveis de ocorrer.

Tendo então o conjunto de todos os movimentos, podemos adotar como critério, escolher aquele com maior *ratio* como sendo o movimento que o usuário irá realizar.

Para o aprendizado da RNA Multi-Camadas, adotamos o algoritmo básico proposto por Rumelhart[6], e objetivando uma maior velocidade, utilizamos um fator *momentum* dinâmico, além de restringirmos o treinamento à somente os exemplos em que a Rede falha. Alterações estas propostas por Allred[7].

III - A HEURÍSTICA

Uma heurística decide qual é a peça que o computador irá mover. Podemos expressa-la através do algoritmo abaixo:

1. Dado um nível de jogo, encontrar todos os movimentos possíveis de serem efetuados pelo computador, os quais inicialmente tem um *ratio* igual a 1;
2. Contar quantos destes movimentos podem capturar peças do adversário;
3. Para cada movimento, fazer :
3.1 Contar quantas peças o adversário pode capturar antes e depois do movimento, com isso podemos saber se este é um movimento que entrega uma peça, defende outra(s), ou nenhuma das anteriores;
3.2 Fazer temporariamente o movimento;
3.3 Estabelecer um *ratio* inicial para o movimento, que será:
   - 0.01 se é possível capturar alguma pedra do adversário e o movimento não o faz (peça será "soprada");
   - 1.00*Número de capturas, se o movimento captura pedra(s) do adversário;
   - 0.2 se o após o movimento, o número de peças que o adversário pode capturar é maior do que antes;

- 0,8 se após o movimento, o número de pedras que o adversário pode capturar for menor do que antes;
- 0,6 se nenhuma das condições anteriores for satisfeita.

3.4 Se o nível de jogo for superior a 1, e o atual *ratio* do movimento for maior ou igual ao maior *ratio* até o momento encontrado, fazer:

- Solicitar da Rede qual será o próximo movimento a ser efetuado pelo usuário;
- Determinar o *ratio* do movimento do usuário, com base no que foi definido em 3.3;
- O novo *ratio* do atual movimento do computador será dado por:

  *ratio_anterior * (1 - ratio_do_usuário)*
- Se o novo *ratio* ainda for igual ou superior ao maior *ratio*, então decrementar o nível em uma unidade, recursivamente determinar o próximo movimento, multiplicar o atual *ratio* pelo o do próximo movimento, e voltar ao passo 3.4 .

4. O movimento a ser executado é aquele que obteve o maior *ratio* dentre todos.

Tal heurística além de permitir vários níveis de jogo, que no caso representam limitações a árvore de busca, ainda se utiliza da Rede Neural para antecipar qual será o próximo movimento do usuário.

IV - A IMPLEMENTAÇÃO

Com o objetivo de validar o modelo proposto, foi implementada uma versão do Jogo, escrita na linguagem PASCAL e utilizando os princípios que regem a programação orientada à objetos.

Nesta implementação, tal qual definido anteriormente, o tabuleiro foi representado por uma matriz 8x8 de números inteiros (curtos). O movimento foi definido como sendo um registro composto pelas coordenadas inicias e finais da peça a ser movida, além do campo *ratio*. A Rede foi definida a partir das matrizes W11, W12 e W2, para os pesos entre a entrada e a primeira camada oculta, entre esta e a segunda camada oculta, e entre a segunda camada oculta e a camada de saída, respectivamente. As entradas para treinamento foram definidas como sendo listas encadeadas de vetores de 64 elementos, enquanto as saídas como listas encadeadas de vetores de 12 elementos. Os conjuntos de movimentos também foram expressos por listas encadeadas.

Foram criados para cada usuário 3 arquivos. O primeiro contendo a Rede Neural do usuário, ou seja, as matrizes W11,W12 e W2 para a Rede treinada com seus movimentos. Outro arquivo que corresponde ao *log* da última partida jogada pelo usuário, sendo formado por pares do tipo **T** e **M**, ou seja, matriz tabuleiro e movimento efetuado. E um último arquivo que contém uma base de dados sobre todos os pares **T,M** do usuário. É importante observar que como para uma mesma matriz **T** podemos ter movimentos **M** diferentes, em cada partida, adotou-se o seguinte critério de seleção:

- Havendo dois movimentos diferentes para uma mesma disposição do tabuleiro, será escolhido aquele que pertence a uma partida na qual o usuário ganhou. Caso isto não ocorra, ou ambos implicaram em vitória ou derrota, os mesmos serão mantidos na base de dados com os seus respectivos *ratio's*, contudo na aprendizagem, o erro permitido para estes será maior.

Quando um usuário joga pela primeira vez com o programa, temos que a Rede ainda "não o conhece", desta forma, a heurística utilizada ao invés de consultar à Rede qual o próximo movimento do usuário, consulta a ela mesma, recursivamente, qual seria seu próximo movimento se ela fosse o usuário. O mesmo ocorre quando a Rede gera movimentos inválidos.

Sempre após o término de uma partida, a partir do arquivo de *log*, é atualizada a base de dados do usuário, e então a Rede é submetida ao processo de aprendizagem da mesma. Com isto, a cada partida, aumenta o "conhecimento" que a Rede possui do usuário, e conseqüentemente, sua capacidade de "antecipar" seus movimentos.

V - CONCLUSÕES

Com o propósito de validar o modelo, a implementação citada foi submetida a testes com diversos usuários diferentes, e observamos:

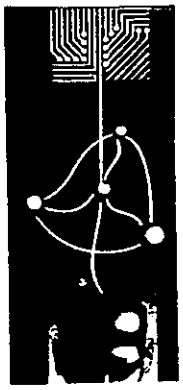
- Inicialmente, isto é, após uma única partida, o índice de acerto da Rede foi em média próximo de 17%;
- Para uma quantidade de movimentos na base de dados acima de 120, algo próximo de 7 partidas, o índice de acerto da Rede foi em média de 83%, não superando este valor;
- Para o caso citado acima, o programa em média ganhou quase todas as partidas disputadas, com raras exceções. Estas observadas quando o usuário apresentava uma grande variação de seus movimentos diante de uma mesma situação.

Os resultados obtidos mostraram-se satisfatórios, vindo de encontro às expectativas. Contudo vale ressaltar trabalhos futuros podem ser desenvolidos tendo em mente as seguintes melhorias:

- Aprimoramento do algoritmo de aprendizagem de forma a obter uma maior velocidade;
- Aprendizado durante o jogo, e não em *batch* como feito atualmente;
- Implementação de outros tipos de Redes Neurais;

VI - REFERÊNCIAS

[1] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K. Lang, IEEE Trans. on Acoustics, Speech, and Signal Processing, ASSP-37, March, 1989.
[2] G.A. Carpenter, S. Grossberg, Appl. Opt., 26, 4919 (1987).
[3] J.J. Hopfield, D.W. Tank, Biol. Cybern. 52, 141 (1985).
[4] R.A. Francelin, F.A. Gomide, K. Loparo, Proc. IJCNN'91 - Int. Joint Conf. on Neural Networks, 3, 2639 (1991).
[5] G.A. Tagliarini, J.F. Christ, E.W. Page, IEEE Trans. on Computers, 40, 12 (1991).
[6] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Parallel Distributed Processing : Explorations in the Microstructure of Cognition", Vol 1, MIT Press, 1986.
[7] L. G. Allred, G. E. Kelly, Proc. IJCNN'89 - Int. Joint Conf. on Neural Networks, 1, 721 (1989).



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

REDES NEURAIS NA QUÍMICA

Aguinaldo Robinson de Souza, Adriana Maria Francisco (Departamento de Química), Leandro Vernaschi de Mello, Rodrigo Alves Marson, Natália Moniwa (Departamento de Computação), Universidade Estadual Paulista, Bauru, S.P., 17030-360.

Resumo

Neste artigo são apresentadas algumas aplicações recentes de modelos de redes neurais na química. A enfase é dada ao algoritmo de "back-propagation" e mapas de Kohonen. As áreas da química que são abordadas são: Química Orgânica, Química Analítica e Química Biológica. São apresentados estudos sobre relações entre a estrutura química de um molécula e seu espectro de infravermelho; predições de reatividade de ligações químicas; estudos sobre a previsão da estrutura secundária de proteínas e estudos sobre relações quantitativas estrutura - atividade biológica.

I - Introdução.

Na química, como em todas as outras ciências naturais, muitos estudos estão sendo feitos para o desenvolvimento de novos métodos, que possam melhorar o tratamento de dados obtidos experimentalmente. A abordagem dos estudos de redes neurais à este problema é básicamente um método que possa tratar sistemas de multivariáveis e multirespostas.

O interesse dos químicos, e também dos bioquímicos, engenheiros químicos e farmacêuticos, no estudo de redes neurais tem crescido rápidamente desde 1986. Este interesse foi despertado em resposta aos trabalhos de Rumelhart e colaboradores (1) e ao aparecimento de um artigo de revisão de Lippmann (2). Os trabalhos referenciados no presente artigo cobrem uma gama de aplicações bastante ampla. O denominador comum destes estudos é a elaboração de modelos. Quando a abordagem de rede neural é utilizada na química, o que geralmente é procurado é algum modelo que possa representar a transformação de um certo tipo de entradas (input) em um conjunto de saídas (outputs). A vantagem de uma rede neural é que esta pode ser utilizada para generalizações (dentro de certos limites).

O modelo geralmente é construído a partir de um banco de dados, por exemplo, um banco de dados em espectroscopia ou a sequência de amino ácidos em proteínas. Este banco de dados deve conter um certo número de correlações, e outras informações que não podem ser deduzidas de uma maneira direta, seja por cálculos teóricos ou métodos numéricos.

II - Correlação estrutura-espectro de infravermelho.

A elucidação da estrututa de compostos orgânicos está bastante relacionada ao métodos espectroscópicos de análise. No entanto, as relações entre a estrutura de uma molécula orgânica e os dados espectrais são geralmente muito complexos para serem expressos em equações explícitas.

Até o momento estas relações foram estudadas utilizando-se regras empíricas. A procura de métodos que implementem o desenvolvimento da relação estrutura-espectro de infravermelho pode, afordunadamente, ser conduzida para o estudo de bancos de dados, de resultados experimentais, computadorizados. Hoje em dia com os resultados experimentais bastante precisos, estes bancos de dados possuem informações muito importantes sobre a estrutura de moléculas orgânicas. Desta maneira, a relação entre estes resultados experimentais e a elucidação da estrutura de moléculas é uma das áreas bastante promissora para a aplicação das técnicas de redes neurais.

Estas relações estrutura-espectro, não está restrita sómente à espectroscopia de infravermelho, mas pode envolver outras técnicas, como por exemplo espectroscopia de massas ou espectroscopia de ressonância magnética nuclear de C^{13} (3,4).

Como um exemplo desta aplicação apresentamos o trabalho de Munk e colaboradores sobre a interpretação de espectros de infravermelho (5). Neste trabalho um intervalo do espectro de 4000 - 400 cm^{-1} foi dividido em 640 intervalos de comprimento 5.6 cm^{-1}. O valor da intensidade de transmissão de um dado intervalo foi dado pela equação:

$$x_i = 1.00 - (\%t) / 100.0 \qquad (1)$$

onde %t = % de transmissão.

Desta maneira a rede neural necessitou de 640 unidades de entrada. No entanto como este valor era muito grande e causou alguns resultados espúrios, este número foi reduzido para 256. Ao mesmo tempo o comprimento dos intervalos foi ajustado de tal maneira que fossem pequenos à baixas baixas frequências e grandes à altas frequências. A expressão que leva em conta a dependência do comprimento do intervalo i com a frequência, é dada pela equação abaixo:

$$i = 6.0\ (\text{frequência})^{0.5} - 120.0 \qquad (2)$$

Esta equação atribui um intervalo de frequência de 10 cm^{-1} (de 400 - 410 cm^{-1}) à unidade de entrada 1. No outro lado do espectro, esta equação atribui um intervalo de frequência de 20 cm^{-1} (de 3928 - 3948 cm^{-1}) à unidade de entrada 256. Estes intervalos de frequência são então distribuidos entre os picos do espectro; se um pico é encontrado, a sua frequência (o valor deve estar entre 0.000 e 1.000) é a entrada desta unidade, senão a entrada desta unidade é igual a zero.

A estrutura do composto é descrita em termos de 36 grupos funcionais: alcool primário, fenol, amina terciária, éster, etc., cada um representado por uma unidade de saída. Desta maneira o vetor a ser encontrado é um vetor binário com 36 variáveis, onde cada número 1 indica a presença do grupo funcional associado, e o número zero indica a sua ausência. Em geral, uma estrutura pode ter vários grupos funcionais, e desta maneira várias unidades de saída podem estar ativas simultâneamente. Após o treino de 14 diferentes redes, variando de 10 até 60 neurônios, 34 foi encontrado como o melhor número.

Devido à possibilidade de uma rápida associação nas redes neurais e também da habilidade em armazenar e tornar acessível grandes quantidades de dados em espaços multidimensionais, Zupan (6) especulou sobre a possibilidade da criação de um espectrômetro inteligente. A sua premissa foi que a quantidade de informações processadas por um espectrômetro de infravermelho, na sua vida média, não excede 1.6 Gbyte (100 dias,

vezes 200 espectros por dis, vezes 4000 valores por espectro, vezes 2 bytes para cada valor). Tal valor está dentro das capacidades de software e hardware existentes hoje em dia.

III - Estrutura secundária de proteínas.

Os polipepitídeos e as proteínas são constituídos de unidades elementares chamados aminoácidos Uma proteína é constituida, salvo alguns casos especiais, de sómente 20 aminoácidos.

Estes aminoácidos estão arranjados sequêncialmente numa proteina; a sequência exata é chamada estrutura primária. Esta sequênica linear enovela-se (folds) em uma única estrutura tri-dimensional, que contém as caracteristicas globais do que é chamada estrutura secundária. Existem três tipos de estruturas secundárias: hélice-α, estrutura-β, e novelo aleatório.

A estrutura secundária de uma proteína é de grande importância para a sua atividade biológica.

Desta maneira, existe muito interesse na predição da estrutura secundária de proteínas a partir do conhecimento de sua estrutura primária. O método mais utilizado é o de Chou e Fasman (7,8), que permite predizer, a partir da sequência de aminoácidos, se um dado aminoácido faz parte de uma α-hélice, uma estrutura-β, ou um novelo aleatório.

Nos últimos anos, muitos estudos foram feitos sobre a utilização de redes neurais para a predição de estruturas secundárias de polipepitídeos a partir da sequência de aminoácidos; os pioneiros neste campo foram Qian e Sejnowski (9).

A suposição básica dos trabalhos de Chou e Fasman e também de Qian e Sejnowski, é de que a identidade de um aminoácido e seus vizinhos determina a estrutura secundária de seus vizinhos. Um certo tipo de "janela" sobre um segmento de um polipepitídeo poderia em princípio fornecer detalhes sobre a estrutura secundária da cadeia inteira.

Na abordagem de redes neurais são necessários três neurônios de saída, um para cada estrutura: hélice-α, estrutura-β, e novelo aleatório. Qian e Sejnowski tentaram diferentes números de neurônios (0 - 80) e decidiram que o número ideal foi 40. O teste de treinamento consistiu de 106 proteínas, tendo um total de 18.105 aminoácidos. Cada um destes esteve acompanhado pela especificação do tipo de estrutura secundária inerente à este resíduo. Um dado aminoácido pode ser encontrado em diferentes tipos de estruturas em diferentes porteínas, ou mesmo em diferentes partes da mesma proteína.

Os trabalhos de Qian e Sejnowski despertaram grande interesse entre os pesquisadores que estudam a estrutura secundária de proteínas. A partir de então, foram publicados um número expressivo de trabalhos nesta área. No entanto, os resultados obtidos até o momento não são conclusivos, e muitos estudos estão sendo feitos neste campo, de interesse cada vez mais crescente entre os pesquisadores.

IV - Reatividade de ligações químicas.

A predição do mecanismo e dos produtos de uma reação química é um dos objetivos principais dos pesquisadores da área de Química Orgânica. Visto que as reações químicas são iniciadas pela quebra de uma ou mais ligações em uma molécula, o conhecimento da reatividade das ligações químicas, isto é das ligações mais fáceis de se quebrar, é indispensável para a predição de reações químicas.

As reações químicas são governadas por processos polares, onde o resultado é uma carga negativa sobre um átomo e uma carga positiva sobre outro átomo. Este processo é conhecido como heterólise.

Simon e colaboradores (10) treinaram uma rede neural para a predição da quebra de ligações em compostos alifáticos. Esta rede neural foi capaz de predizer com grande exatidão qual a ligação

que se quebraria mais fácilmente, e como as cargas iriam se distribuir entre os átomos da molécula.

Um banco de dados de 29 moléculas foi escolhido como um grupo representativo das variações estruturais do grupo de moléculas alifáticas; estas moléculas possuem 385 ligações capazes de 770 modos potenciais, de quebra heterolítica. Considerando sómente as ligações simples (por exemplo: C-H em um grupo metílico) temos 373 modos diferentes de quebra. Destes 373 modos, uma série de 149 modos foram selecionados e divididos em 43 reativos e 106 não reativos. A quebra de uma ligação química é influenciada por uma variedade de efeitos energéticos, eletrônicos ou estéricos, como por exemplo: distribuição de cargas, indução, ressonância, polarizabilidade, energia de dissociação de ligações, etc.

Dentre os fatores que governam uma reação química temos: a diferença na carga total, Δq_{tot}, a diferença na carga-π, Δq_{π}, a diferença na eletronegatividade-σ, $\Delta \chi_{\sigma}$, a medida da polaridade da ligação, Q_{σ}, a estabilização por ressonância, $R^{\cdot}$, a polarizabilidade da ligação, α_b, e a energia de dissociação da ligação, BDE. Os valores destas variáveis são calculados e atribuídos às ligações usando um programa chamado PETRA (Parameter Estimation for the for Treatment of Reactivity Applications) (11).

O objetivo principal nestes estudos é relacionar estas variáveis á uma classificação de reatividade de uma ligação química. A classificação da reatividade de ligações químicas é claramente um problema de muitas variáveis, e portanto bastante adequado para ser estudado utilizando os modelos de redes neurais.

A primeira abordagem ao problema é classificar os modos de quebra como reativos ou não reativos através de uma rede neural de duas camadas, utilizando o aprendizado por "back-propagation". O número de unidades de entrada é definido como sendo igual a sete, o número de efeitos controladores da reatividade. Estes possuem diferentes valores, um deles possue valores no intervalo de -0.2 a +0.2, outro de 200 a 500; visto que as unidades de entrada esperados devem estar entre 0 e 1, cada unidade de entrada deve ser escalonada (scaled) entre os seus valores máximos e mínimos. A classificação das unidades de saída é: 0 para modos não reativos, e 1 para modos reativos.

O estudo deste sistema foi realizado utilizando uma rede de Kohonen (9x9), contendo 81 neurônios que podem ser considerados como "sub-espaços". Quando o banco de dados possue ligações com uma depência similar sobre as sete variáveis de controle, esta rede irá mapear então estas ligações sobre o mesmo neurônio.

A rede é estabilizada após 30 ciclos de treinamento, isto é, após os 373 modos de quebra de ligação terem sidos analisados na rede por 30 vezes. Seis neurônios estão vazios, 56 possuem modos classificados e 19 estão ocupados por modos não classificados. A rede de Kohonen utiliza uma técnica de aprendizado não supervizionada visto que não utiliza a informação da classe quando está no processo de aprendizado. Esta rede possui uma quantidade de informação, de interesse dos químicos, muito grande.

Encontrou-se que os modos reativos (**R**) formam um agregado no centro do mapa. Esta é uma indicação de uma auto-organização durante o processo de aprendizado da rede de Konohen, que percebe a similaridade entre certos tipos de modos e coloca-os então no mesmo neurônio. Esta rede também leva ao reconhecimento da similaridade de todos os modos reativos colocando-os em neurônios vizinhos, formando desta maneira agregados de neurônios com modos reativos.

V - Relações Quantitativas Estrutura - Atividade Biológica (QSAR).

O campo de estudo de relações quantitativas estrutura - atividade biológica

foi introduzido no início dos anos 60 com os trabalhos pioneiros de Hansch e colaboradores (12,13). Em uma sequência de publicações estes pesquisadores demonstraram que a atividade biológica de certos compostos químicos obedece a uma função matemática das suas características fisico-químicas, como por exemplo: hidrofobicidade, forma e propriedades eletrônicas. Estes métodos estão sendo largamente adotados pelas indústrias farmacêuticas e agroquímicas (14). A pesquisa de tais relações é uma das aplicações mais importantes das técnicas de modelagem molecular.

A correlação da estrutura química de fármacos com suas atividades biológicas é de particular interesse, devido principalmente ao elevado custo de novos fármacos. Uma predição quantitativa, confiável, de sua atividade antes do novo fármaco ser sintetizado é de grande interesse para os laboratórios farmacêuticos que realizam estas sinteses.

Podemos citar alguns trabalhos que mostram a relevância do tema. Temos o trabalho de Marzona (15) sobre complexos de inclusão de esteróides com ciclodextrinas. A inclusão de uma molécula de fármaco na ciclodextrina pode alterar consideravelmente suas características, principalmente no plano farmacotécnico (16).

Os estudos de Tetko e colaboradores (17), sobre a aplicação de redes neurais artificiais no estudo de derivados de carboquinonas, também mostram a crescente importância destes modelos no estudo de QSAR. Os derivados das carboquinonas são amplamentes utilizados como agentes anti-tumor, por exemplo contra a leucemia L-1210 (18).

Como um exemplo típico da aplicação de modelos de redes neurais em QSAR discutiremos o trabalho de Aoyama e colaboradores (19,20).

O banco de dados neste estudo envolveu modificações no esqueleto básico da carboquinona. Muitas carbiquinonas exibem graus variados de atividade anticarcinogênica. Este estudo de QSAR teve como objetivo predizer a dose mínima de fármaco necessária para produzir uma extensão de 40% na vida de cobaias, ratos que foram inoculados com células de leucemia L-210.

Esta dose efetiva mínima depende da concentração, C, da substância necessária para obter o efeito necessário, e é representada como log (1/C). Quanto mais efetiva for o fármaco menor será a concentração necessária.

Como era esperado encontrou-se que a atividade anticarcinogênica depende da identidade dos substituintes R^1 e R^2, na estrutura da carboquinona. Nas análises de regressão multilinear, correntemente usadas, estes substituintes são descritos por variáveis fisico-químicas que descrevem a influência combinada dos substituintes R^1 e R^2. A atribuição dos substituintes como R^1 e R^2 é baseada nas suas refratividades molares: $MR_1 \leq MR_2$. Neste estudo foram utilizados onze diferentes substituintes R^1 (consistindo básicamente de grupos alquila de cadeia curta como o grupo metila, etila e propila) e cerca de 30 diferentes substituintes R^2, estes de cadeia mais longa e carregando funcionalidades adicionais, como $-CH_2CH_2OCH_3$ e $-CH(OCH_3)CH_2OCONH_2$.

A rede consistiu de seis unidades de entrada e um neurônio de saída. Após várias tentativas 12 neurônios foram necessários na "hidden layer". Esta rede de (6 x 12 x 1) neurônios foi treinanda com 35 carboquinonas utilizando o algorítmo de "back-propagation". Os valores de log (1/C) foram então comparados com os resultados obtidos através de análise de regressão multilinear, com as mesmas 35 carboquinonas.

A atividade anticarcinogênica de 17 das carboquinonas foi predita com uma

maior precisão do que o estudo utilizando análise de regressão multilinear; para 6 compostos os resultados tiveram a mesma qualidade, e para 12 os resultados foram piores. De uma maneira geral os resultados utilizando redes neurais são significativamente (mas não dramáticamente) melhores do que a análise de regressão multilinear.

Aparentemente o problema estudado é bastante adequado para ser tratado por modelos lineares, mas a abordagem utilizando redes neurais pode ir um pouco mais além. Problemas de QSAR não lineares poderão ser melhores estudados quando modelados por redes neurais.

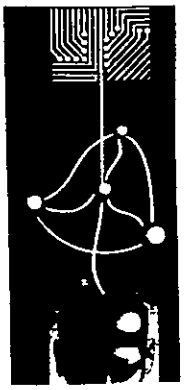
VI - Conclusões e perspectivas.

Uma das consequências das aplicações de redes neurais na química é de que somos forçados a reconsiderar o modo como representamos e interpretamos os dados obtidos experimentalmente. A representação dos dados é muito importante para a extração de informações. A abordagem de redes neurais, seja no aprendizado via mapas de Kohonen ou "back-propagation", colocou este fato em mais evidência ainda; a representação dos dados é crucial.

Acreditamos que a tendência nesta área de pesquisa será a de desenvolvimento de soluções dedicadas. Desta maneira os laboratórios industriais irão procurar patentear tais soluções, como já vem acontecendo na indústria farmacêutica, e a troca de informações sobre as pesquisas nesta área serão desta forma afetadas. Portanto, torna-se cada vez mais importante que os órgão governamentais, através de auxílios às Universidades, suportem as pesquisas nesta área.

VI - Referências Bibliográficas.

1) D. E. Rumelhart, G.E. Hinton, R.J. Willians, *Microstructures of Cognition, Vol I, MIT Press*, Cambridge, (1988).
2) R. P. Lippmann, *IEEE ASSP Magazine*, April 4 (1987).
3) B. Curry, D.E. Rumelhart, *Tetrahedron Comput. Methodol.* **3** (1990) 213.
4) V. Kvasnicka, *J. Math. Chem.* 6 (1991) 63.
5) M.E. Munk, M.S. Madison, E.W. Robb, *Mikrochim. Acta [Wien]* II (1991) 505.
6) J. Zupan, *Anal. Chim. Acta*, **53** (1990) 53.
7) P.Y.Chou, G.D. Fasman, *Biochemistry* 13 (1974) 211.
8) P.Y. Chou, G.D. Fasman, *Biochemistry* 13 (1974) 222.
9) N. Qian, T.J. Sejnowski, *J. Mol. Biol.* 202 (1988) 865.
10) V. Simon, J. Gasteiger, J. Zupan, *J. Am. Chem. Soc.*, in press.
11) J. Gasteiger, M. Marsili, M.G.Hutchings, H. Saller, P. Löw, P. Röse, K. Rafeiner, *J. Chem. Inf. Comput. Sci.* **30** (1990) 467.
12) C. Hansch,R.M. Muir, T. Fujita, P. Maloney, E. Geiger, M. Streich, *J. Am. Chem. Soc.* **86** (1964) 2817.
13) C. Hansch, T. Fujita, *J. Am. Chem. Soc.* **86** (1964) 1616.
14) J.E. Ridings, D.T. Manallack, M.R. Sauders, J.A. Baldwin, D.J. Livingstone, SmithKline Beecham Pharmaceuticals, Welwyn, AL6 9AR, Herts, England, *Quant. Struct.-Act. Rel.* **12** (1993) 272.
15) M. Marzona, R. Carpignano, P. Quagliotto, *Quant. Struct.-Act. Rel.*12 (1993) 299.
16) A. Korolkovas, *Rev. Bras. Med.* 49 (1992) 509.
17) I.V. Tetko, A. I. Luik, G. I. Poda, *J. Med. Chem.* **36** (1993) 811.
18) M. Yoshimoto, H. Miyazawa, H. Nakao, K. Shinkai, M. Arakawa, *J. Med. Chem.* 22 (1979) 491.
19) T. Aoyama, Y. Suzuki, H. Ichikawa, *J. Med. Chem.* **33** (1990) 905.
20) T. Aoyama, Y. Suzuki, H. Ichikawa, *J. Med. Chem.* **33** (1990) 2583.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajuba. 24 a 27 de outubro de 1994

PLANEJAMENTO DE TELECOMUNICAÇÕES: AGRUPAMENTO (CLUSTERING) DE CENTRAIS USANDO REDES NEURAIS

Walmir Matos Caminhas. DEE. UFMG. Hermano M.F. Tavares. F.E.E. . UNICAMP
E-MAIL: DEPTO @ DENSIS. FEE. UNICAMP. BR
DENSIS - FEE /UNICAMP - Caixa Postal 6101
13081-970 - Campinas - SP

RESUMO

O presente trabalho fornece uma alternativa para a "clusterização" de centrais telefônicas. usando uma abordagem baseada em redes neurais artificiais. A **rede neural** utilizada é do tipo auto-organizável. com duas camadas de neurônios. É utilizado o aprendizado por competição para ajustar os pesos sinápticos da rede. na fase de treinamento. O programa de simulação foi testado para duas configurações de redes. uma contendo 12 centrais e outra 26. Os resultados obtidos utilizando tal abordagem são de boa qualidade. quando comparados com resultados obtidos com métodos heurísticos clássicos de "clusterização.

1- INTRODUÇÃO

A tecnologia das telecomunicações sofreu um grande impacto. nos últimos 15 anos. devido aos avanços da eletrônica digital de alta velocidade e o barateamento das fibras ópticas. Estes fatores estão proporcionando uma ampla modernização dos sistemas de telecomunicações. que caminha em ritmo apressado em termos de digitalização e opticalização dos equipamentos de comutação e roteamento. bem como dos meios de transmissão.

O uso constante de técnicas de informática no mundo das telecomunicações abre caminho para a incorporação de serviços telemáticos a preços atraentes. o que provoca um expansivo aumento no volume do tráfego.

Estes dois fatores. a inovação tecnológica e o aumento do tráfego. tornaram antiquados os métodos tradicionais de planejamento e estão exigindo novas ferramentas computacionais [1] que utilizam modelos mais aderentes aos novos equipamentos e suas particularidades evolutivas.

2 - TELECOMUNICAÇÕES: PLANEJAMENTO

O planejamento de redes de telecomunicações se efetua. basicamente em três fases: planejamento da rede externa; planejamento da comutação e planejamento da rede de transmissão. Na primeira fase é feita a localização e o dimensionamento das centrais. providenciando um balanço apropriado entre custos dos equipamentos de comutação e custos da rede de acesso. Na segunda são estabelecidos os planos de encaminhamento e dimensionamento de troncos entre as centrais. Finalmente. na terceira fase é feito o dimensionamento dos equipamentos roteadores.

roteamento de fibras ópticas e de outros meios de transporte e é definida a rede de galerias a utilizar.

Em relação ao Planejamento da Transmissão, a tendência é concentrar grandes fluxos de informação num número reduzido de enlaces, favorecendo economias de escala. A expectativa é obter soluções com grandes profusão de ligações lógicas em malha assentadas em uma estrutura física pouco malhada. Esta aparente contradição é resolvida pela existência de "rótulas" (roteadores) inteligentes distribuídas em pontos estratégicos do sistema. A fase do Planejamento da Transmissão é dividido em cinco etapas [2]:

- **agrupamento** das centrais formando "Clusters";
- **enfeixamento** da demanda, onde se dimensiona os equipamentos roteadores;
- seleção das **galerias** candidatas a receber fibras ópticas;
- **roteamento** dos enlaces;
- **evolução** da situação atual à situação futura planejada.

A primeira etapa é resolvida por heurística clássicas [3,4] de "clusterização" e tem por objetivo disciplinar a rede de telecomunicações para aplicações dos métodos utilizados nas etapas seguintes.

O presente trabalho fornece uma alternativa para a "clusterização" em questão usando uma abordagem baseada em redes neurais artificiais.

3 - AGRUPAMENTO (CLUSTERING)

3.1 - O MODELO DA REDE DE TELECOMUNICAÇÕES

A rede de telecomunicação é descrita por um grafo $G(\mathbb{N},\mathbb{A})$, sendo que $\mathbb{N}$ e $\mathbb{A}$ representam os conjuntos de nós (centrais) e arcos (galerias), respectivamente. Os arcos são adirecionais e caracterizados por um comprimento c_{ij}. A cada par de nós $(i,j) \in \mathbb{N}^2$ associamos também uma grandeza escalar d_{ij}, a demanda de troncos de telecomunicação, considerada como um dado do problema.

A partir da rede definida acima, o objetivo do trabalho é agrupar subconjuntos de nós (centrais) de acordo com grau de afinidade, definido a partir das demandas e distâncias entre eles. O grau de afinidade entre os nós pode ser medido a partir da matriz de similaridade $S_{(n\times n)}$ definida pela quádrupla $(\mathbf{C}, \mathbf{D}, \theta_1, \theta_2)$, sendo $\mathbf{C}_{(n\times n)}$ a matriz de distâncias entre nós, $\mathbf{D}_{(n\times n)}$ a matriz de demanda, θ_1 e θ_2 são escalares positivos e n é cardinalidade do conjunto $\mathbb{N}$. Como o objetivo é obter subconjuntos (cluster) com alta demanda interna e com nós próximos, usando a abordagem de rede neural, um elemento s_{ij} da matriz de similaridade é calculado por:

$$S_{ij} = \theta_1 \cdot (1/d_{ij}) + \theta_2 \cdot c_{ij} \qquad (1)$$

No processo de "clusterização" que será adotado cada cluster é formado a partir de um nó semente chamado de "hub", que terá a função de agregar e rotear as demandas entre clusters. Com isto, a rede de transporte pode ser dividida em duas subredes: rede intracluster e rede intercluster.

3.2 - UMA ABORDAGEM USANDO REDES NEURAIS ARTIFICIAIS

Ultimamente, Redes Neurais Artificiais tem recebido uma grande atenção como ferramenta computacional em diversas áreas, entre as quais o processamento digital de sinais e imagens, reconhecimento de padrões, controle de processos, "clusterização" e outras [5].

A topologia da rede muito usada para determinar formação de cluster é mostrada na figura 1 e é a utilizada neste trabalho. Na figura temos que os neurônios da primeira camada são os de entrada e os da segunda os de saída. A matriz de pesos sinápticos $\mathbf{W}$ é composta pelos vetores coluna $[\mathbf{W}^1\ \mathbf{W}^2\ ..\ \mathbf{W}^j\ ..\ \mathbf{W}^m]$. Estes vetores coluna, bem como o vetor de entrada $\mathbf{X}$ são normalizados.

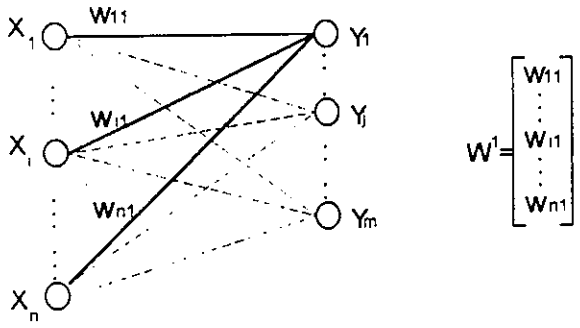


Fig. 1 - Topologia da Rede Neural Utilizada.

3.3 - O ALGORITMO DE APRENDIZADO (COMPETIÇÃO).

PASSO 1: A partir das matrizes de demanda **D** e distâncias **C** e dos parâmetros θ_1 e θ_2 monte a matriz de similaridade **S**. Para obter as distâncias entre nós não vizinhos, pode ser utilizado o algoritmo de caminho mínimo de Djkstra [6]. Para montar **S** as matrizes **C** e **D** devem possuir colunas normalizada (norma quadrática igual a 1).

PASSO 2: Inicialize a matriz de pesos sinápticos **W** aleatoriamente. Normalize todas as colunas de **W**.

PASSO 3: Gere um número **i** aleatoriamente, onde **i** $\in \mathbb{N}$ (corresponde ao índice de um nó da rede de telecomunicações). Monte o vetor de entrada da rede neural igual à coluna **i** da matriz de similaridade. Provoque uma perturbação em todos os elementos de **X**. Normalize o vetor **X**.

PASSO 4: A partir de **X** e da matriz de pesos sinápticos **W** determine o neurônio vencedor, aquele cuja distância entre o vetor coluna $\mathbf{W^j}$ (coluna **j** da matriz **W**) e **X** é a menor. Seja **L** o índice do neurônio vencedor. Na iteração k, cujo o vetor de entrada é representado por **X**(k), apenas os pesos da coluna **L** da matriz **W** ($\mathbf{W^L}$) são ajustados. Para este ajuste, temos duas situações:

1 - Se $\mathbf{i} \notin \mathbb{H}$, sendo $\mathbb{H}$ o conjunto dos hubs, definido como $\mathbb{H} = \{ h_j / h_j \in \mathbb{N}, j=1, 2 \ldots m\}$. Neste caso temos:

$$W^L(k+1) = \frac{W^L(k) + \gamma\left(X(k) - W^L(k)\right)}{\left\| W^L(k) + \gamma\left(X(k) - W^L(k)\right)\right\|} \qquad \text{sendo } 0 < \gamma < 1 \qquad (2)$$

2 - Se $\mathbf{i} \in \mathbb{H}$ e $\mathbf{i}=h_L$, a correção se faz utilizando a mesma expressão. Caso contrário, isto é se $\mathbf{i} \neq h_L$, troca-se o sinal de γ na expressão.

PASSO 5: Repetir os passos 3 e 4 até o número de amostra do treinamento.

PASSO 6: Após o treinamento, com os pesos já atualizados determinar os neurônios vencedores para todos os nós da redes, obtendo-se assim os subconjuntos de nós com um certo grau de afinidade (cluster).

4 - APLICAÇÃO

O procedimento acima foi aplicado para determinar os clusters das redes mostradas nas figuras 2 e 3. Todas as distâncias apresentadas são em quilômetros. Os "hubs" e os clusters obtidos são destacados na figura. As tabelas 1 e 2 mostram as demandas para as duas redes consideradas, as matrizes de demanda são simétricas. Para efeito de cálculo da matriz de similaridade, demandas nulas foram substituídas por um valor pequeno.

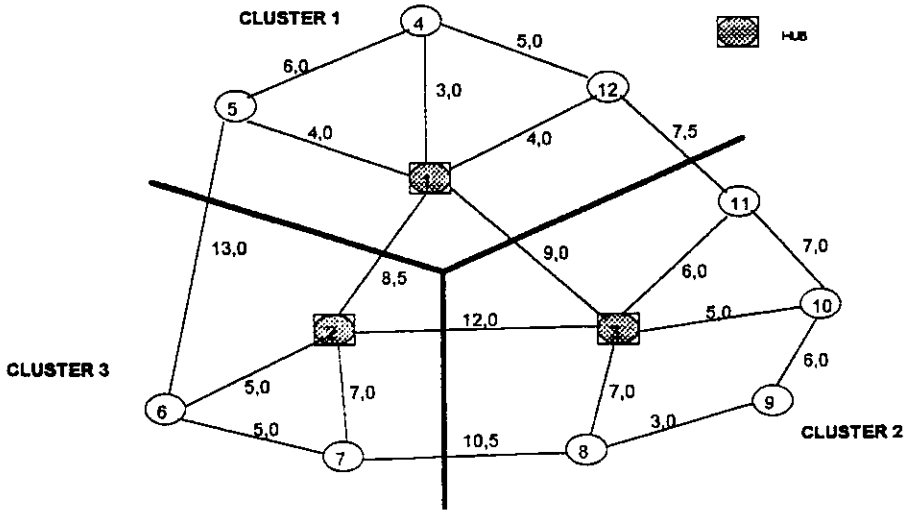


Fig. 2 - Configuração da Rede de 12 centrais

Tabela 1 - Demandas das Rede de 12 Centrais

$$\begin{bmatrix}
0 & 20 & 30 & 30 & 50 & 26 & 10 & 8 & 6 & 16 & 10 & 60 \\
 & 0 & 10 & 16 & 28 & 34 & 38 & 18 & 16 & 24 & 20 & 14 \\
 & & 0 & 10 & 12 & 14 & 14 & 40 & 50 & 32 & 30 & 10 \\
 & & & 0 & 22 & 32 & 18 & 14 & 8 & 26 & 20 & 36 \\
 & & & & 0 & 22 & 10 & 14 & 14 & 10 & 18 & 26 \\
 & & & & & 0 & 40 & 24 & 20 & 20 & 24 & 20 \\
 & & & & & & 0 & 18 & 16 & 12 & 20 & 10 \\
 & & & & & & & 0 & 18 & 40 & 36 & 4 \\
 & & & & & & & & 0 & 34 & 26 & 8 \\
 & & & & & & & & & 0 & 38 & 6 \\
 & & & & & & & & & & 0 & 16 \\
 & & & & & & & & & & & 0
\end{bmatrix}$$

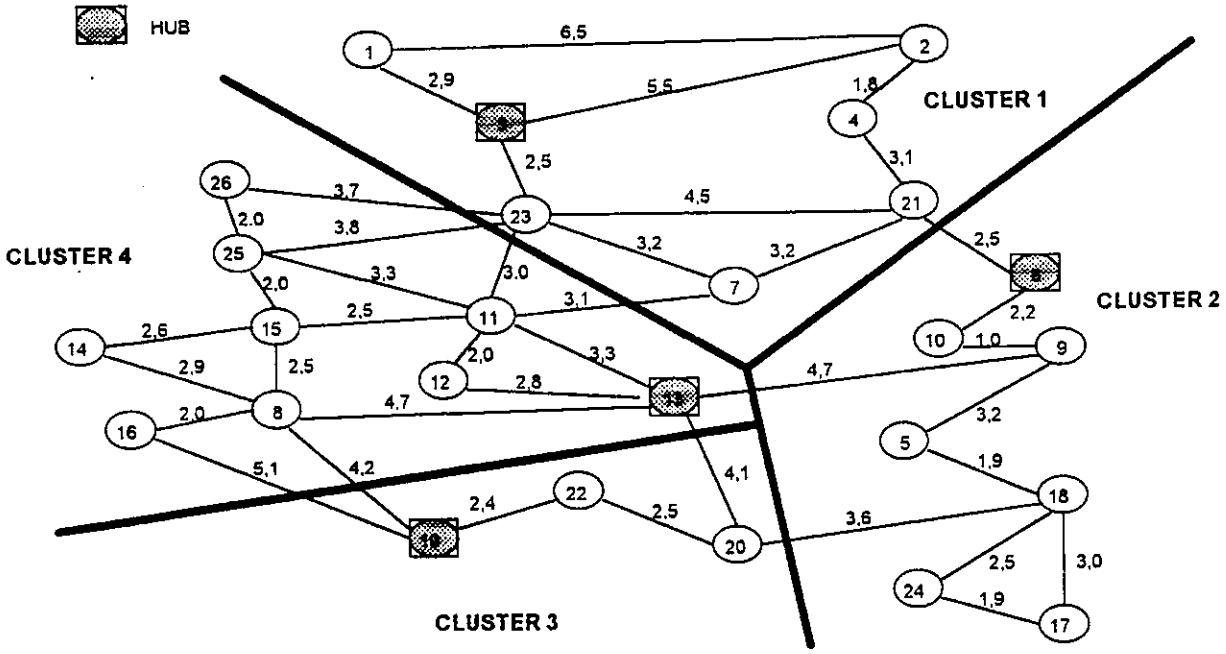


Fig. 3 - Configuração da Rede de 26 centrais

Tabela 2 - Demandas da Rede de 26 Centrais

0	29	8	19	0	1	5	1	15	3	1	30	1	0	1	0	0	5	0	1	4	2	4	0	1	2
	0	48	34	0	9	24	6	0	11	18	0	5	0	6	0	0	26	2	2	19	10	15	0	3	7
		0	14	0	2	6	1	12	03	2	49	1	0	2	1	0	2	1	0	6	0	11	0	3	7
			0	0	7	18	5	35	14	22	0	8	0	3	1	0	10	1	3	20	1	11	0	2	6
				0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
					0	16	6	45	38	45	41	14	0	8	1	3	44	2	7	13	3	9	1	3	5
						0	12	0	40	31	0	0	1	15	1	5	45	6	12	26	3	18	0	6	8
							0	36	23	32	41	27	26	13	5	3	32	13	12	4	3	13	1	3	4
								0	27	0	42	42	3	26	3	8	0	22	35	38	43	27	12	13	25
									0	42	07	49	3	14	4	8	20	10	30	18	2	21	3	9	15
										0	49	03	7	37	8	3	1	7	16	18	13	23	5	11	11
											0	42	7	37	10	0	0	0	0	0	0	0	0	32	34
												0	16	19	5	9	46	15	22	10	5	14	1	9	10
													0	1	0	0	2	1	1	0	1	0	0	0	0
														0	2	4	22	8	10	5	0	11	1	7	7
															0	1	3	3	1	1	1	01	0	1	1
																0	30	4	0	0	6	0	6	0	0
																	0	30	0	24	8	37	4	5	8
																		0	9	2	31	2	3	2	2
																			0	9	45	4	8	1	1
																				0	4	29	1	6	10
																					0	1	0	0	1
																						0	0	28	18
																							0	0	0
																								0	8
																									0

5 - CONCLUSÃO

Os resultados apresentados foram comparados com outros obtidos utilizando-se métodos heurísticos clássicos [2], e fica evidenciado que a técnica de redes neurais é uma boa alternativa para "clusterização" de centrais telefônicas. A vantagem desta técnica é que uma vez treinada a mesma rede pode ser usada para avaliar pequenas alterações de projeto. Um ponto importante que está sendo incorporado no programa é a possibilidade de se limitar o número máximo de centrais em cada cluster, para aplicações práticas desta abordagem. Outro aspecto a considerar seria a possibilidade de só aceitar um neurônio como vencedor caso seja atendida uma restrição de vizinhança entre o hub e a central em questão, de forma a não desorganizar a rede telefônica sob outras óticas (manutenção, tarifação, etc..) alheias à nossa formulação. Seria também importante testar alternativas quanto ao cálculo da matriz de similaridade.

6 - REFERÊNCIAS BIBLIOGRÁFICAS

[1] Wu, T. H. (1992). Fiber Network Service Survivability, **Artech House**.

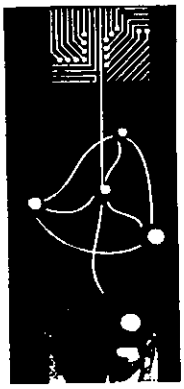
[2] Metodologia de Planejamento da Transmissão - OS - No. 08 - Relatórios Técnicos do ContratoTELESP/ UNICAMP 1560/93.

[3] SÁ LUCAS, L. C. (1983). "Análise de Grupamento"- Dissertação de Mestrado UFRJ . pp 1-77

[4] KLINCEWICZ, J.G. (1991). "Heuristics for then p-Hub Location Problem", European Journal of Operational Research, No. 53, pp 25-37.

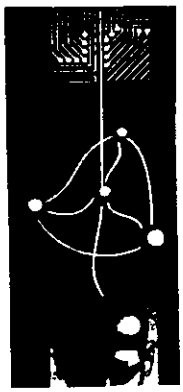
[5] KOSKO, B. (1992). Neural Networks and Fuzzy Systems, **Prentice Hall**.

[6] HILLIER, F.S.; LIEBERMAN, G.J. (1990). Introduction to Operations Research, **McGraw-Hill Publishing Company.**



MISCELÂNEA II

Anotações



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

APLICAÇÃO DE REDES NEURAIS EM PREVISÃO HIDROLÓGICA

RUBENS ALMIRON (1)

RESUMO

Este trabalho apresenta aplicações práticas das redes neurais no campo da hidrologia, realizadas tentando obter soluções adequadas aos problemas já estudados com outras ferramentas.

INTRODUÇÃO

Os esforços realizados para aplicar a tecnologia computacional a imagens, discursos e previsões compartem algumas caraterísticas.

Os mesmos se baseiam no reconhecimento de padrões, definido como habilidade de identificar e classificar eventos ou sucessos apesar do ruído e da distorção.

Deve-se considerar o êxito comercial que estas intenções poderiam produzir, pois elas são um reflexo do mundo do homem.

As redes neurais aparecem, neste contexto, como uma resposta clara às necessidades, já que permitem, no geral, aumentar a precisão na solução de problemas até hoje sem resposta e diminuir o custo destas soluções.

Uma rede neural consiste em um algoritmo inspirado em estudos do funcionamento do cérebro humano. Logicamente, que estas redes neurais em seu funcionamento têm pouca relação com a biologia.

A tecnologia das redes neurais baseia-se no aprendizado que os computadores realizam diretamente dos dados solucionando problemas de classificação, estimação e compreensão de dados e outras tarefas similares.

Conhecidas por décadas estas redes como soluções em busca de problemas,

(1) Engenheiro Hidrólogo da Divisão de Hidrologia (OPSH.EO) da ITAIPU BINACIONAL

ultimamente, portanto, começam a aparecer em aplicações praticas e esta tendência só pode acelerar agora com o "hardware" especializado disponível.

Centenas de aplicações atuais usam redes neurais, tais como FAX providos com **OCR (Optical Character Recognition)**, sistemas eletrocardiográficos, controle de processos, sistemas financeiros, reconhecimento de alvos e controle de vôos assim como sistemas de comunicações.
As redes neurais aparecem como resposta à necessidade de reconhecimento de padrões, gerada desde o momento em que os computadores começam a interatuar com o mundo real.

CONCEITOS

As redes neurais são constituídas por nós, semelhantes a neurônios humanos, que recebem informações e a passam mediante conexões ponderadas (Ver figura 1).

Estas redes aprendem variando os pesos de ponderação de tais conexões. Uma rede neural, com pesos adequados, é capaz de modelar qualquer função computável.
Um nó de uma rede neural normalmente recebe informação de entrada (dado) a multiplica por um peso correspondente à conexão que recebeu o dado, efetua a somatória destes produtos e processa esta somatória mediante uma função de transferência não linear produzindo um resultado. Então, a parte fundamental destas redes é este processo de composição do mencionado produto e a velocidade computacional depende da eficiência com que realiza os produtos e acumulações.

As redes neurais estão compostas por camadas de nós. Como mínimo uma rede neural simples possue três camadas de nós. A primeira é a camada de nós de entrada, a segunda denomina-se camada de nós ocultos e a terceira camada de nós de saída. Na primeira camada tem-se um nó por cada dado de entrada (Ver figura 2).

A primeira camada recebe a informação e a encaminha aos nós ocultos que se conectam, finalmente, com os nós de saída. Os nós ocultos e os nós de saída recebem a informação realizando os processos de multiplicação e acumulação aplicando uma função de transferência para produzir um resultado.

Uma vez que a informação tenha sido conduzida até os nós de saída, estes produzem um resultado que é comparado com o valor esperado (valor observado) encontrando a diferença entre estes valores. Desta maneira, os nós de saída encontram o vetor de erros e ao mesmo tempo elaboram as derivadas das componentes do vetor em função dos respectivos pesos e o transferem, agora no sentido inverso, voltando atrás, aos nós ocultos (esta operação dá o nome à rede denominada **BACK PROPAGATION NETWORK**). Os nós ocultos calculam a somatória das derivadas dos erros definindo assim sua participação no erro de saída.

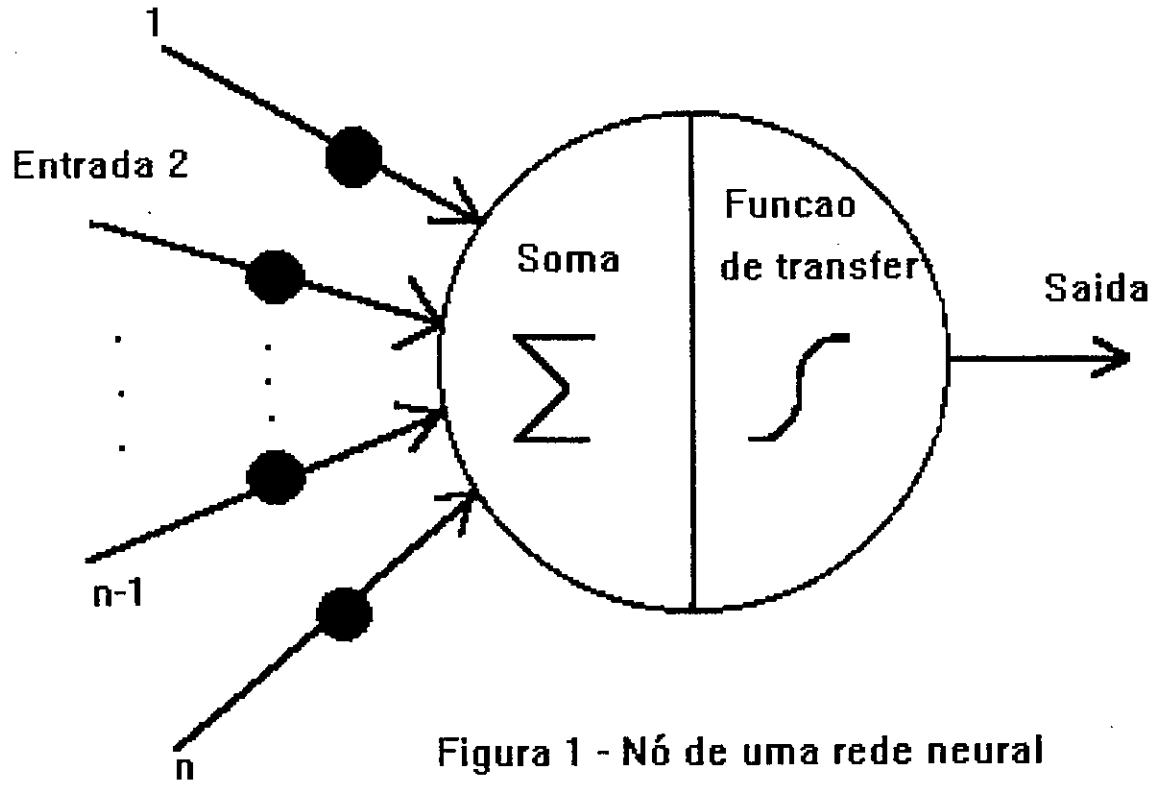


Figura 1 - Nó de uma rede neural

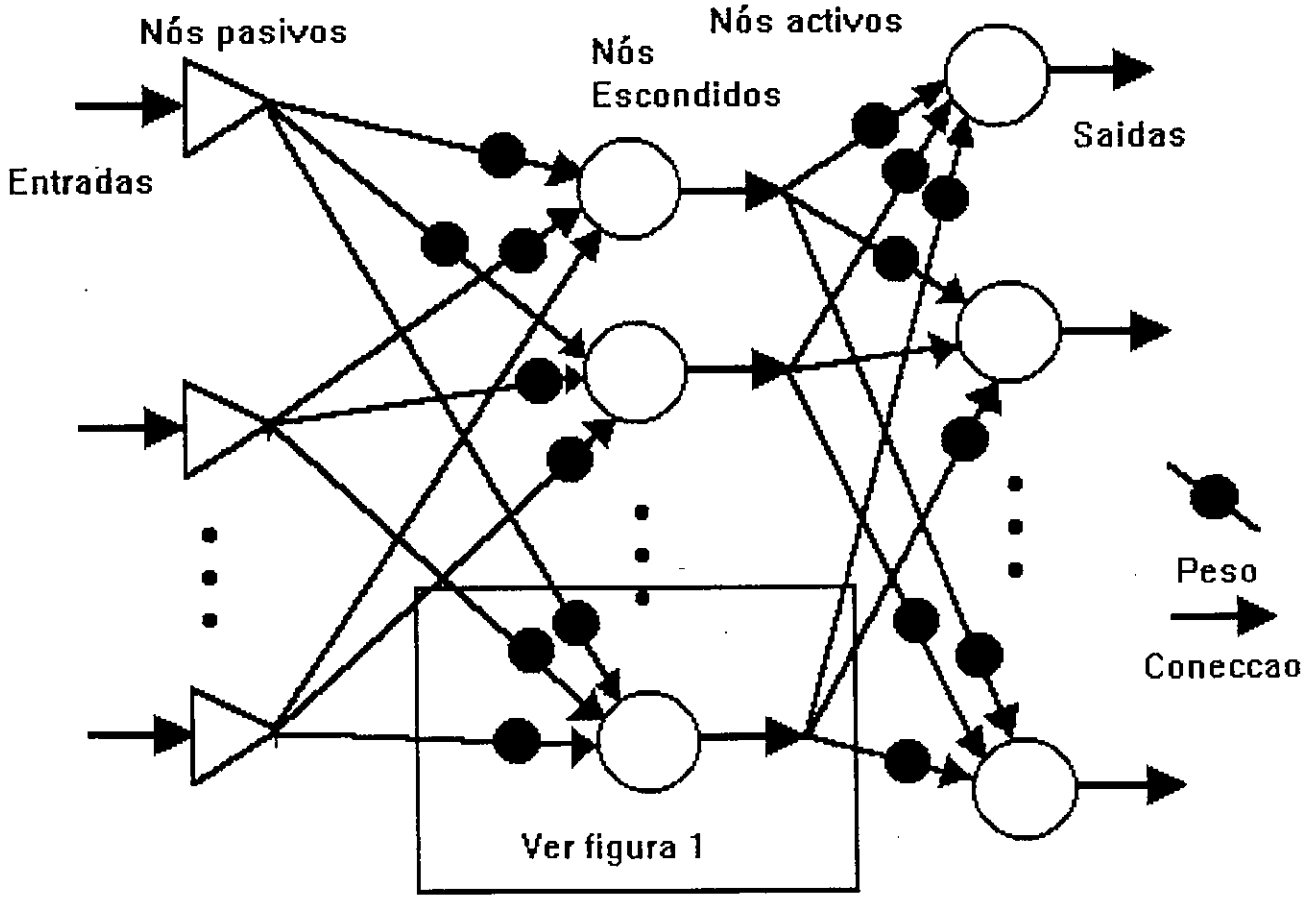


Figura 2 - Rede neural elemental

Tanto os nós ocultos como os de saída realizam a modificação dos pesos das respectivas conexões mediante algum procedimento matemático previamente determinado, por exemplo mínimos quadrados, diminuindo seu erro.

Este processo repete-se ciclicamente até que os erros dos nós de saída, sejam, todos, menores que um valor previamente estipulado.

APLICAÇÕES DE REDES NEURAIS

Uma aplicação de reconhecimento de padrões que usa uma rede neural pode ser classificada em termos de como emprega a rede. As de maiores categorias cobrem o uso de uma rede neural para classificação ou estimação de funções e compressão de dados, extração das caraterísticas e ordenamento estatístico. Cada uma delas permite uma aplicação específica. Uma rede neural classificadora combinada com uma camara, por exemplo, pode eleger papas, ler códigos de correio ou encontrar um rastro digital em uma base de dados. Combinada com um microfone, pode diagnosticar problemas mecânicos, identificar sons embaixo d'água ou gerar sinais para eliminar vibrações.

BENEFÍCIOS

São incalculáveis em diversos aspectos. Primeiro, são adaptáveis: podem receber dados e aprender a partir deles. Por tanto, inferem soluções desde os dados iniciais capturando relações muito ocultas.

Esta habilidade difere radicalmente das técnicas de "software" normais porque não dependem do conhecimento prévio das regras que governam um fenômeno estudado pelo programador. Podem reduzir o tempo de desenvolvimento aprendendo relações subjacentes que sem elas são difíceis de se encontrar e descrever. Podem, também, resolver problemas que escapam às soluções existentes.

Segundo, podem generalizar: conseguem processar corretamente dados que só guardam leves relações com os dados nos quais foram originalmente treinadas. A generalização é útil porque os dados do mundo real estão carregados de elementos distorcidos.

Terceiro, as redes neurais são não-lineares, com o que podem captar interações complexas entre as variáveis de entrada de um sistema. Em um sistema linear, a mudança de um dado de entrada produz uma variação proporcional na saída, e os efeitos dos dados dependem só de seus próprios valores. Por outro lado em um sistema não-linear os efeitos dependem de outras entradas e as relações são função de ordem superior.

Os sistemas do mundo real são, a princípio, não lineares. Na economia de um país, por exemplo, os preços, taxas de juros, nível de emprego e outros fatores interagem entre si. O efeito da mudança de preço depende,

por exemplo, das taxas de juros de tal modo que as mesmas mudanças produzem diferentes efeitos em diferentes condições. As redes neurais oferecem um meio de solução prática para tais sistemas complexos.

Quarto, são altamente paralelas: suas numerosas, idênticas e independentes operações podem ser executadas simultaneamente. Os processadores paralelos podem executar centenas ou até milhares de vezes mais rápido que os convencionais microprocessadores e processadores de sinais digitais. Agora que existem computadores e "chips" paralelos de propósitos especiais para aplicações de redes neurais, portanto redes grandes podem alcançar velocidades de tempo real e inclusive produtos da vida diária podem empregar redes. Este incremento em velocidade e economia fazem praticas muitas aplicações pela primeira vez, proporcionando desenvolvimentos futuros.

DESVANTAGENS

O conhecimento das redes neurais assim como a experiência humana baseiam-se em processos muito difíceis de explicar, entretanto, confia-se no critério de uma pessoa com suficiente conhecimento sobre o tema, que não é a mesma situação que ocorre com os resultados produzidos por uma rede, existe uma grande reação de desconfiança sobre o desconhecido.

O processo de aprendizagem imperfeitamente compreendido é outro obstáculo para aplicação das redes.

Finalmente, a necessidade de contar com tempo de computação prolongado para realizar o aprendizado mediante computadores muito eficientes é a terceira desvantagem.

APLICAÇÕES PRÁTICAS NO CAMPO DA HIDROLOGIA

Até este momento tem-se analisado dois problemas bem definidos que não haviam sido satisfatoriamente resolvidos mediante métodos matemáticos conhecidos.

Eles referem-se, especificamente, a previsão de vazões em bacias de escoamento rápido com base em informações de precipitação e de ocorrência de vazões anteriores e a previsão de níveis no canal de fuga da UHE Itaipu sujeita a influência das vazões do rio Iguaçu, das descargas da própria usina e o estado do nível anterior.

PREVISÃO DE VAZÕES EM BACIA RÁPIDAS

O caso estudado refere-se a bacia do rio Chopim, afluente do rio Iguaçu a jusante da UHE Salto Osório, com uma área de drenagem da ordem de 6.696 km² em relação à estação de Águas do Verê.

Utilizou-se dados dos anos de 1982 e 1983 de vazões e precipitações. A precipitação média na bacia foi

definida mediante o método de Thiessen baseado nas estações pluviométricas de Águas do Verê, Palmas, Salto Claudelino e Pato Branco no período do estudo de 01/01/1982 até 13/07/1983.

Como entradas foram considerados quinze nós nos quais introduzem-se os seguintes dados:

Vazões de 5 dias anteriores ao dia atual;
Precipitações de 5 dias anteriores ao dia atual; e
Precipitações de 5 dias posteriores ao dia atual, incluindo este.

Como saídas foram utilizados as cinco vazões correspondentes aos cinco dias seguintes, incluindo o dia atual como sendo o primeiro deles.

Para o aprendizado da rede usaram-se 477 dias e para comprovação utilizaram-se 100 dias. Na figura 3 a seguir, inclue-se um gráfico que representa os valores observados e simulados obtidos para os 100 dias mencionados.

PREVISÃO DE NÍVEIS NO CANAL DE FUGA DA UHE ITAIPU

Este problema tem sido estudado utilizando outras metodologias como regressões de diferentes tipos sem encontrar-se uma solução razoável para o mesmo.

As variáveis que configuram esta situação são as descargas da UHE Itaipu, as vazões do rio Iguaçu e o estado anterior de níveis d'água no canal de fuga. O estudo realizou-se em base horária neste caso.

Nesta ocasião usaram-se 51 nós de entrada que representam aos seguintes dados:

Descargas de 48 horas anteriores à hora atual; e
Níveis d'água no posto fluviométrico, denominado Capanema no rio Iguaçu.

Os nós de saída representam os níveis no canal de fuga correspondentes aos níveis 24 horas posteriores à hora atual incluindo esta.

Na figura 4, apresenta-se os resultados obtidos.

CONCLUSÕES

À vista dos resultados obtidos conclui-se que esta metodologia aplicada às previsões hidrológicas apresenta caraterísticas extremamente promissoras. Portanto recomenda-se, no âmbito atual, a continuação das análises, inclusive estendendo o alcance dos problemas analisados para incluir outros de interesse da área de hidrologia, assim como aqueles relativos a áreas de operação e supervisão que apresentem caraterísticas afins para a aplicação das redes neurais.

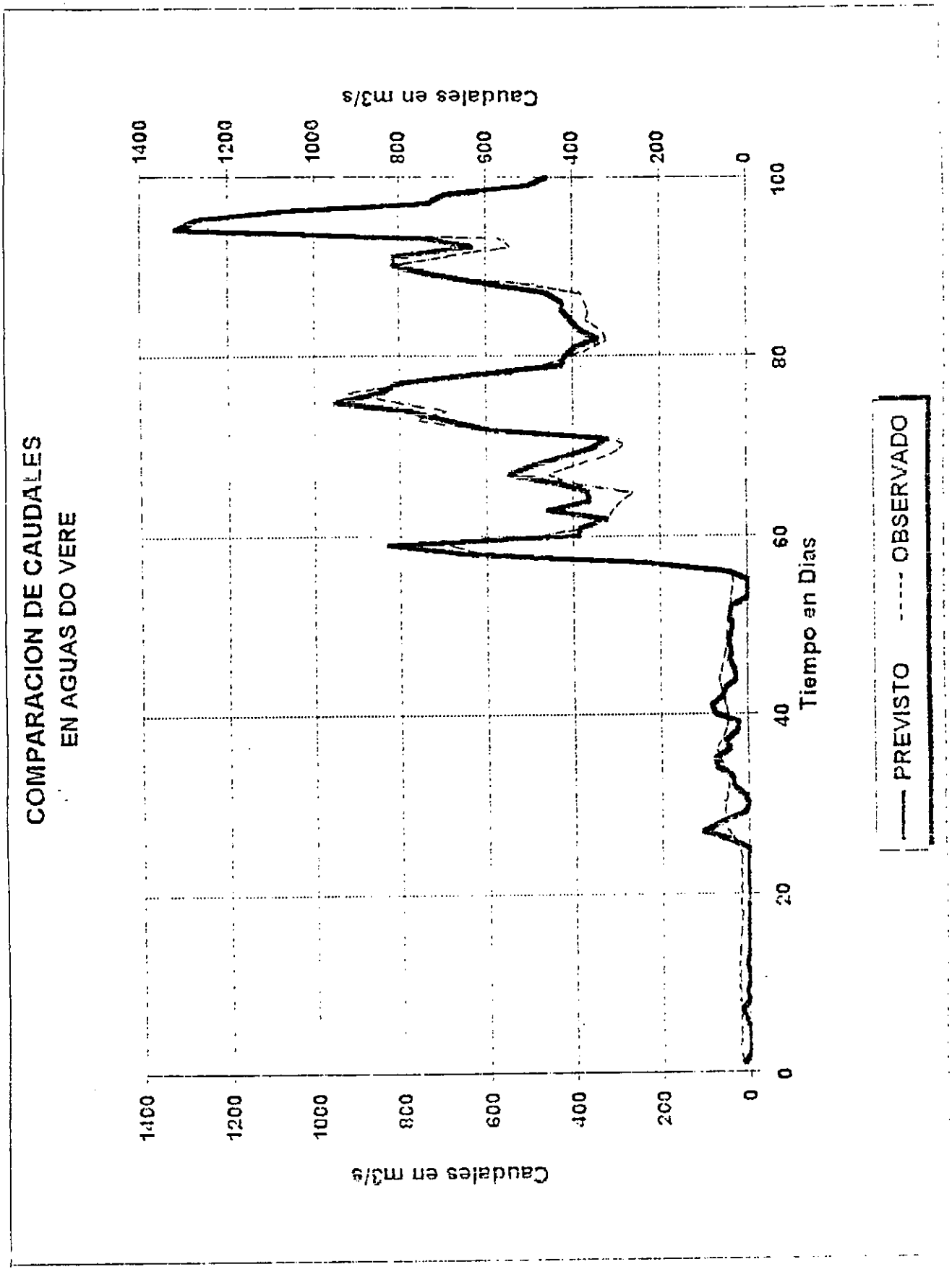

COMPARACION DE CAUDALES
EN AGUAS DO VERE
Caudales en m3/s
1400
1200
1000
800
600
400
200
0
0
20
40
60
80
100
Tiempo en Dias
PREVISTO
OBSERVADO

FIGURA 3

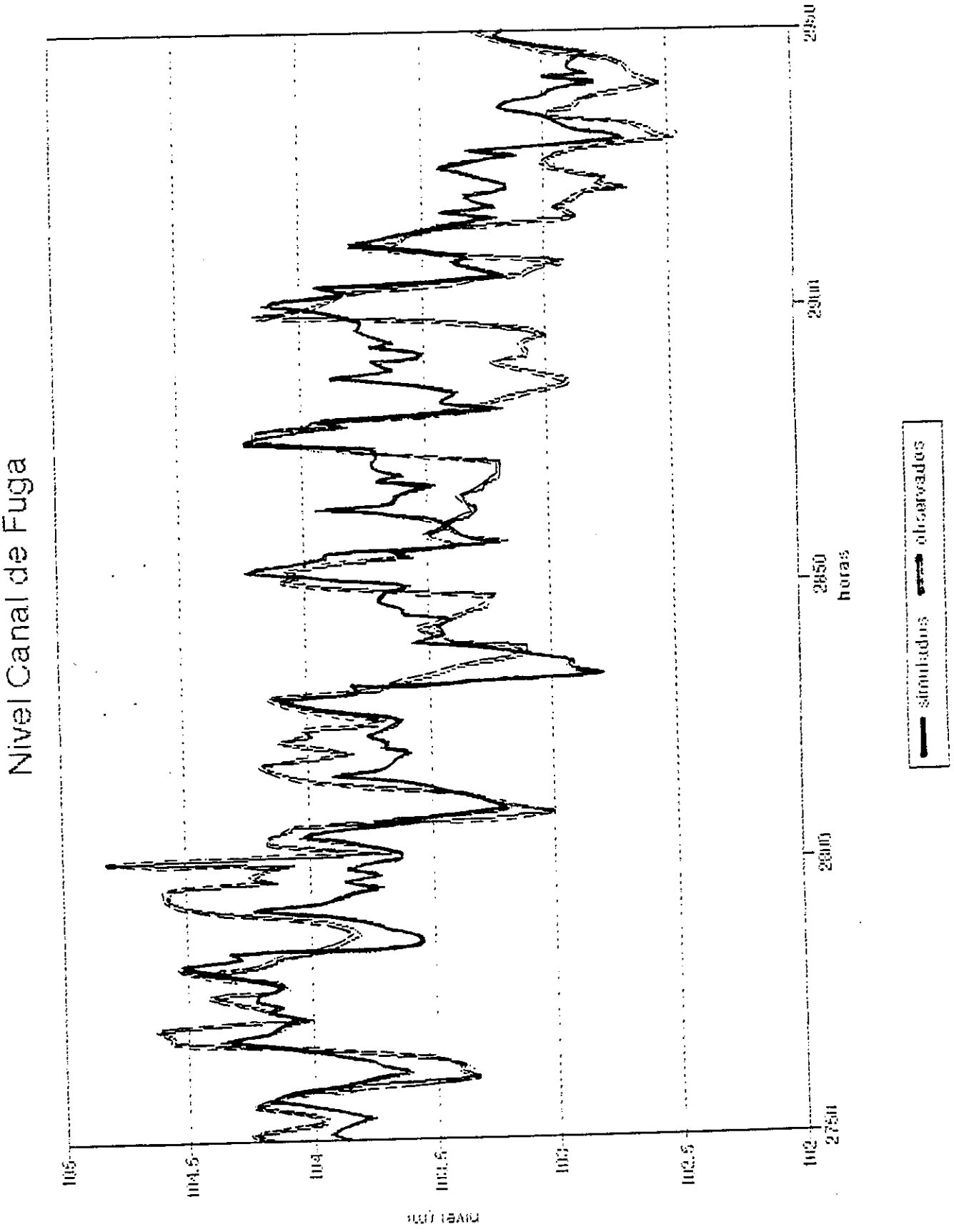

Nivel Canal de Fuga
105
104.5
104
103.5
103
102.5
102
2750
2800
2850
2900
2950
horas
simulados
observados

FIGURA 4

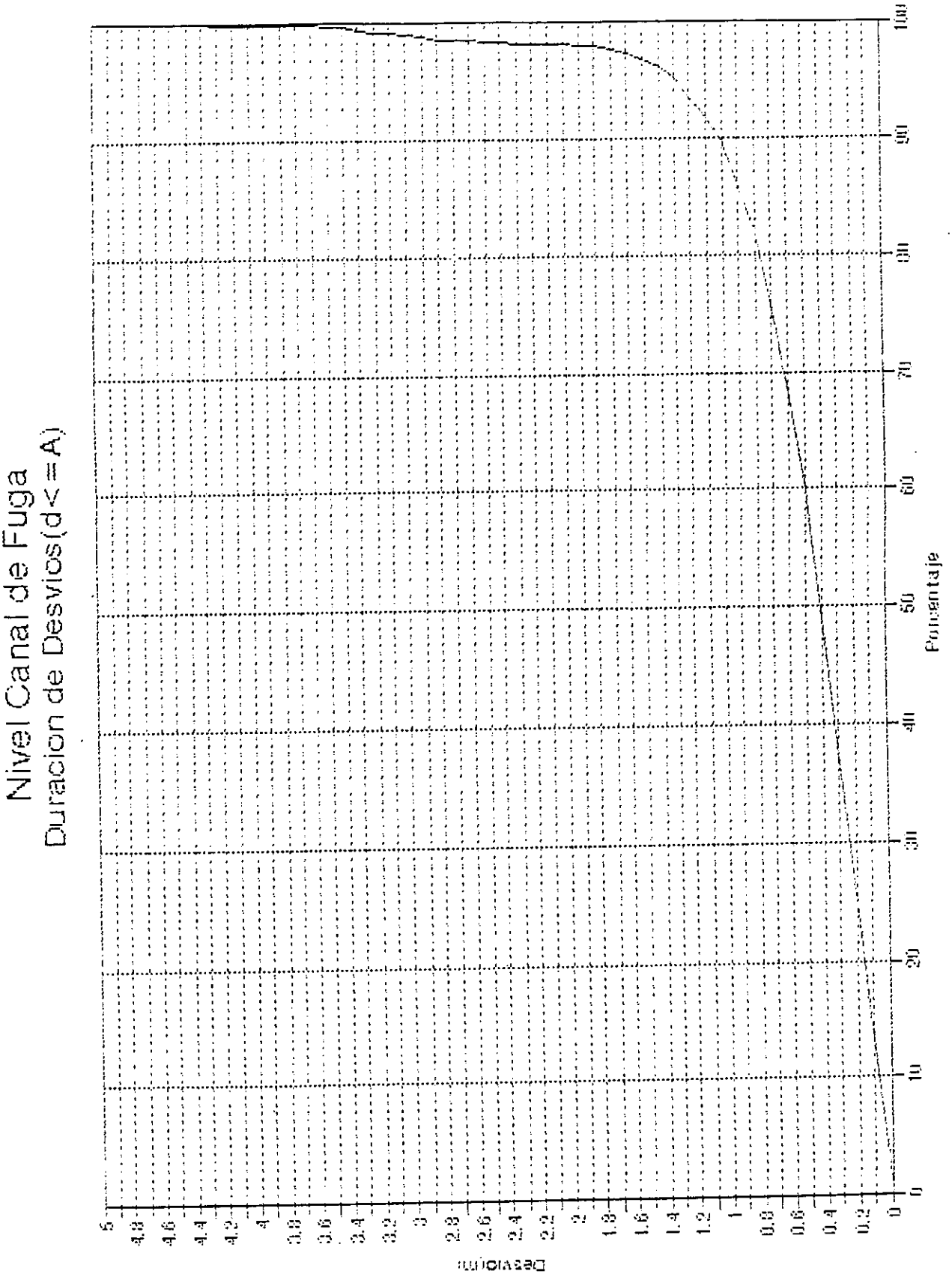
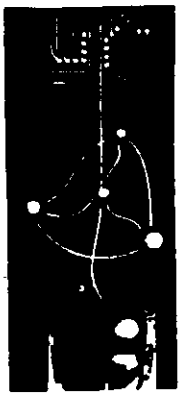

Nivel Canal de Fuga
Duracion de Desvios(d<=A)
Desviacion
Porcentaje
0
10
20
30
40
50
60
70
80
90
100
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3
3.2
3.4
3.6
3.8
4
4.2
4.4
4.6
4.8
5

FIGURA 5



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajuba
Itajuba. 24 a 27 de outubro de 1994

Modelagem de Superfícies Usando Redes Neurais

Luiz Carlos da Silva
Clylton Galamba Fernandes
Edson Costa de Barros Carvalho Filho

UFPE-Universidade Federal de Pernambuco
Departamento de Informática
Cx 7851. 50.732-970. Recife. PE. Brasil
{lcds@di, cjgf, ecdbcf}@di.ufpe.br

Sumário. Este trabalho apresenta uma técnica para modelagem de superfícies utilizando Redes Neurais Artificiais. visando à geração de objetos físicos sólidos. Será discutida a viabilidade do uso dos Mapas de Kohonen nesta modelagem. É apresentada uma análise de alguns experimentos do ponto de vista da convergencia da rede. observando os melhores parametros usados no processo de treinamento. Será averiguado também os aspectos visuais da superfície gerada: suavidade e distribuição de pontos.

1 Introdução

Em muitas aplicações. as Redes Neurais Artificiais tem sido usadas como ferramenta para reconhecimento de padrões [7]. Elas possuem a habilidade de generalização e adaptação para novas situações. onde após o treinamento de alguns exemplos uma rede é capaz de definir uma representação dos estímulos de entrada nas suas estruturas internas.

A Modelagem de Sólidos [3]. por sua vez. emergiu rapidamente como um dos principais campos de pesquisa e desenvolvimento em diversas área de aplicação. tais como: engenharia e design de produtos. prototipagem eletronica. *motion planning* e *computer-aided manufacturing*. Todas essas aplicações requerem representações de formas de objetos físicos sólidos. É essa modelagem que dá subsídios para geração dessas representações bem como operações primitivas sobre elas.

A Modelagem Geométrica ou de Superfície. subárea da Modelagem de Sólidos. tradicionalmente identifica um conjunto de técnicas que pode modelar certas classes de *piecewise parametric surfaces* [3]. sujeitas a condições particulares de forma e suavidade. Ela tem se desenvolvido como um campo a parte em várias indústrias. incluido a automobilística. a aeroespacial e a de construção civil. Como a modelagem de sólidos empenha-se em envolver completamente a geometria dos objetos. há uma necessidade crescente de se pesquisar a geração. de maneira rápida e simplificada. das formas de superfícies e tecnicas para sua manipulação.

Neste trabalho será investigado não só a modelagem de superfícies a partir de objetos geradores. mais também mecanismos para a manipulação dessas superfícies. criando interativamente novos objetos. Neste sentido utilizou-se o modelo neural proposto por Kohonen em [6]. O estudo da viabilidade dessa modelagem é realizada analisando a visualização do comportamento dessas redes durante seu processo de aprendizagem. Isto é feito devido à rede poder ser interpretada como um *wireframe* [2]. grade de arame. da superfície de um sólido. onde cada um dos seus neuronios representa um ponto no espaço euclidiano[1].

Este trabalho é apresentado em quatro seções. onde esta é a primeira. Na segunda seção será exposto como é feita essa modelagem. fazendo considerações a respeito da adaptação da rede e visualização da superfície. São mostradas ainda. as razões do uso de Redes Neurais e especificamente dos Mapas de Kohonen[2] para essa modelagem. Na seção tres. é feito um estudo de caso de alguns experimentos realizados. Esta análise aborda tanto o aspecto de convergencia da rede quanto o visual. Na última seção tem-se as conciderações finais juntamente com os trabalhos futuros.

2 Modelagem usando Redes Neurais

Para se conseguir a modelagem de sólidos. foram utilizadas as Redes de Kononen com vetores de peso tridimensionais. onde estes vetores são tratados como pontos. Isto é feito no intuito de usar a matriz de neuronios como um wireframe auto-adaptativo. Considerando a rede como uma grade de pontos será possivel a visualização gráfica da mesma. Os padrões de treinamento da rede também são tratados como pontos. só que estes são originários da superfície de um objeto gerador. Durante o processo de treinamento. esses padrões são apresentados à rede. fazendo com que ela

[1] Para uma maior clareza. neste trabalho utilizar-se-á o termo *ponto* significando *ponto no espaço euclidiano*.

[2] Os termos Mapas de Kohonen e Redes de Kohonen serão usados neste trabalho como sinônimos.

aproxime seus pesos às entradas, gerando, deste modo, uma superfície que representa os padrões de entrada nas suas estruturas internas.

2.1 Adaptação da Rede

A adaptação da rede é feita usando o algoritmo de aprendizagem proposto por Kohonen em [6], o qual usa um processo de aprendizagem não-supervisionado para gerar *clusters*[3], a medida que os padrões vão sendo apresentados à rede. No início desse processo, a rede possui os vetores de peso aleatórios e normalizados com valores variando no intervalo $[-1, 1]$. Tal rede apresenta-se graficamente, neste estágio, como um emaranhado de linhas (Figura 1). No decorrer do processo, entretanto, a rede vai tomando a forma da superfície do sólido gerador através desse processo adaptativo.

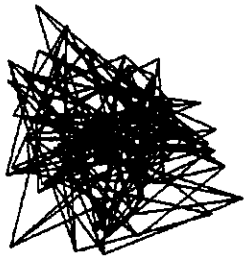


Figura 1: Rede com Pesos Aleatórios

Esse algoritmo de aprendizagem usa o conceito de *Vizinhaça Topológica* [6] que é uma fronteira dinâmica a qual define os nós que envolverão o *neurônio vencedor* e serão estimulados pelo pradrão de entrada corrente. Este limite e a forma como ele é diminuido são particularmente responsáveis pela apresentação final da rede, pois eles permitem a formação e determinam a disposição dos clusters na rede, definindo a forma final da superfície. No início do processo de aprendizagem esse limite é grande chegando a conter a maioria dos neurônios da rede. A medida que o processo avança, entretanto, o raio da vizinhança decresce até conter apenas o neurônio vencedor.

Outro conceito usado no algoritmo de treinamento é o de *Neighborhood Kernel*, função definida sobre a grade de pontos que especifica a distribuição do estímulo sobre os neurônios da vizinhança topológica. Foi utilizado duas definições para essa função: *Borbulhamento* e *Gaussiana* [4]. No borbulhamento todos os neurônios da vizinhança topológica são estimulados igualmente. Já a outra definição, usa uma função gaussiana no cálculo do ajuste dos pesos. Assim os nós mais próximos do neurônio vencedor serão mais estimulados que aqueles da periferia da vizinhança topológica.

Foi escolhida a forma de grade quadrada (Figura 2) para a disposição dos neurônios na rede, ou seja, cada neurônio está ligado graficamente aos seus quatro vizinhos imediatos, exceto os neurônios localizado nas bordas e nas quinas da rede os quais possuem três e dois vizinhos respectivamente. Embora cada neurônio possa ser conectado graficamente a no máximo quatro vizinhos, cada neurônio vencedor está virtualmente ligado e pode disparar todos aqueles que estão dentro de sua vizinhança topológica.

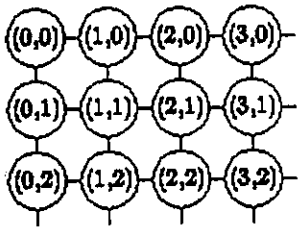


Figura 2: Grade Quadrada

2.2 Visualização da Superfície

A visualização da rede é realizada ligando todos os vetores de peso das linhas e os das colunas da matriz, isto é, conecta-se os vetores de peso dos neurônios de cada linha dois a dois, onde o primeiro é ligado ao segundo, o segundo ao terceiro e assim por diante até o último neurônio de cada linha. O mesmo procedimento é feito para conectar os vetores de peso das colunas, de modo que no final desse processo tem-se uma grade de pontos interconectados, representando uma superfície.

A visualização da rede tridimensional no monitor de vídeo é baseada nos mesmos princípios usados por pintores e designers. A diferença é que o computador usa um modelo matemático em vez da tela de pintura. Assim, para um maior realismo da imagem, foram usadas sobre os vetores de peso da rede as seguintes transformações geométricas tridimensionais: projeção, mudança de escala, rotação sobre os eixos cartesianos e translação de coordenadas.

2.3 Viabilidade do Modelo de Kohonen para Modelagem de Superfícies

As Redes Neurais foram escolhidas para dar suporte a modelagem de superfícies devido suas características intrínsecas de paralelismo e generalização. Pois com a capacidade de adaptação a novas situações

[3] Classes de padrões.

pode-se pensar na rede como um wireframe que se modifica a medida que os padrões vão sendo-lhe apresentados. Esta constante modificação gera uma superfície cujo conjunto de pontos converge para os padrões de treinamento. Já o paralelismo permite que essa convergência seja feita de modo muito rápido.

Optou-se pelas Redes de Kohonen para a implementação desta modelagem devido sua arquitetura apresentar uma matriz de neurônios. Quando se usa vetores de peso tridimensionais esta matriz se assemelha muito a um wireframe.

As características de auto-organização e mapas ordenados também foram levados em consideração nessa escolha: a primeira permite o surgimento de clusters na rede, criando regiões de concentração de pontos na superfície; já a segunda faz com que não haja mudanças abruptas na disposição dos pontos da superfície modelada, pois os clusters variam gradativamente pela rede, apresentando uma topologia mais ou menos ordenadas.

3 Resultados Obtidos

Os resultados obtidos neste trabalho foram originários tanto da análise de convergência da rede como da análise do aspecto visual do wireframe representado pela rede. No primeiro caso foram investigados os melhores valores dos parâmetros para um processo de treinamento eficiente. Já no segundo caso, o interesse maior estava no resultado gráfico final da rede. Embora estes dois aspectos estejam intimamente ligados, nesta secão, será dado as duas abordagem separadamente, fazendo as analogias entre elas a medida que se forem fazendo necessárias.

3.1 Análise da Convergência da Rede

Os parâmetros analisados nos experimentos foram: o *raio inicial* da vizinhança topológica, o *número de apresentações* de cada padrão e o *número de neurónios* da rede.

Na primeira série de experimentos foram utilizados, como conjunto de treinamento, 154 pontos uniformemente espaçados da superfície de um esfera de raio 30. Variou-se o raio inicial da vizinhança topológica e fixou-se o número de nós da rede em 1444 (matriz 38x38), a taxa de aprendizagem em 85% e o número de apresentação em 65, resultando em 10010 iterações.

A variação do Erro Médio em função do raio da vizinhança topológica, usando o processo de treinamento *borbulhamento*, é mostrado na Figura 3, onde cada curva do gráfico representa um experimento começando com um raio inicial indicado nas legendas.

Usando um raio inicial grande (diâmetro da rede) verificou-se que os neuronios passam algumas iterações oscilado entre os padrões de treinamento. Com o passar do tempo, no entanto, o raio diminui e os clusters vão sendo formandos. No final do processo a rede apresentou um erro médio de 3.09, que é razoável haja vista o usado em [4] o qual fica em torno de 3.61.

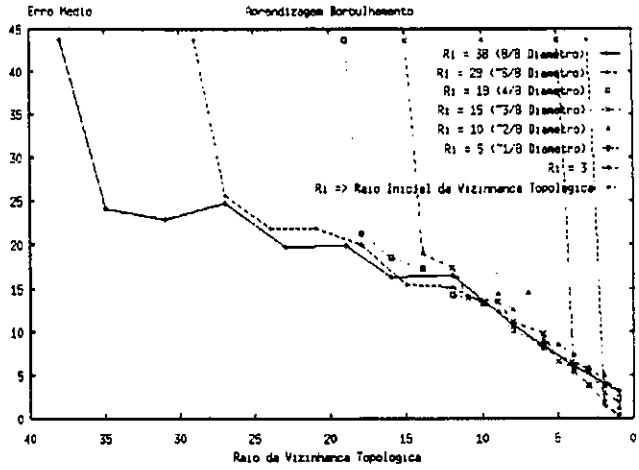


Figura 3: Variação do Erro Médio em Função do Raio da Vizinhança Topológica

O melhor erro médio encontrado, usando o borbulhamento, foi 0.066 com um raio inicial igual a 3 (10% do diametro da rede). Entretanto gerou-se clusters com baixo grau de generalização formados por apenas alguns neuronios. Além disso houve um grande número de nós não comprometidos, 111 *uncommitted neurons*, assim como uma desordenação dos clusters na rede. Neste momento a rede se apresentou muito facetada, assemelhando-se a um cristal, com uma concentação de pontos no seu centro representando os neurônios não comprometidos (Figura 4).

Figura 4: Rede com o Menor Erro Médio

Fez-se necessário a análise não só do erro médio mais também do grau de generalização dos clusters (concentração de pontos), do número de neuronios não comprometidos e a ordenação dos clusters na rede (suavidade da rede).

Fazendo-se essa análise mais completa, a rede que melhor se adaptou aos padrões de entrada foi usando o raio inicial igual a 15 (40% do diâmetro da rede). Neste experimento todos os neurônios foram comprometidos e a rede apresentou um erro médio de 1.79 .

Usando o método *gaussiano* a melhor aproximação para a rede foi obtida usando o raio inicial igual a 5 (13% do diâmetro da rede). Neste experimento todos os neurônios foram comprometidos. Os clusters gerados, entretanto, não atingiram uma ordenação adequada, resultando em uma superfície muito acidentada. A Figura 5 mostra um gráfico do erro médio em função do número de interações para alguns experimentos empregando esse método. Em cada experimento foi usado o raio inicial indicado na legenda.

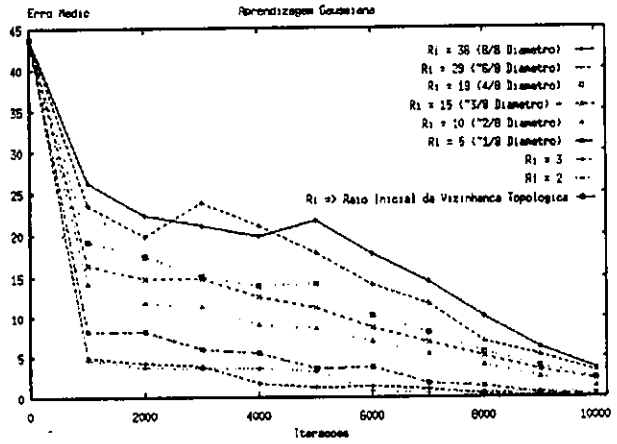


Figura 5: Variação do Erro Médio em Função do Tempo (Método *Gaussiano*)

Na segunda bateria de testes foram fixados o raio inicial em 19 (1/2 diâmetro da rede), a taxa de aprendizagem em 85%, o diâmetro da rede em 38 (matriz $38x38$) e variou-se o *número de apresentação* de cada padrão à rede. O aumento do número de apresentação, resultando um acréscimo do número de iterações, fez com que o erro médio fosse diminuido cada vez mais. O melhoramento desse resultado, porém, está relacionado com o tempo que a rede leva para convergir e em alguns casos este tempo pode ser muito grande para pequenos aperfeiçoamentos.

Na figura 6 é mostrado o gráfico da variação do erro médio em função do tempo (iterações) usando o método *borbulhamento*. O número de apresentações é mostrado nas legendas.

Na última série de testes da convergencia foi analisado a *quantidade de neurônios* da rede. Para isso, foram fixados o número de apresentação em 65, o raio inicial na metade do diâmetro da rede e taxa de adap-

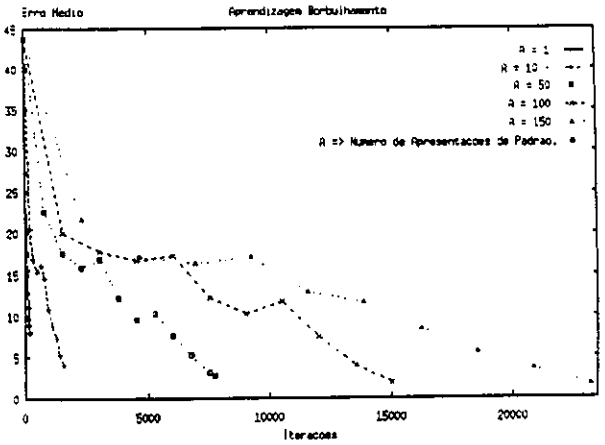


Figura 6: Variação do Erro Médio em Função do Tempo (Método *Borbulhamento*).

tação em 85%. A medida que aumentava-se o número de neurônios na rede o erro médio também era reduzido e os clusters eram melhor definidos gerando uma superfície menos acidentada. Não foi registrado o aparecimento de neurônios não comprometidos.

A figura 7 mostra o comportamento do erro médio final da rede em relação ao número de neuronios dos clusters, isto é, o numéro de total de neurônios dividido pela quantidade de padrões. O gráfico mostra duas curvas, uma usando o método *Borbulhamento* e outa o *Gaussiano*.

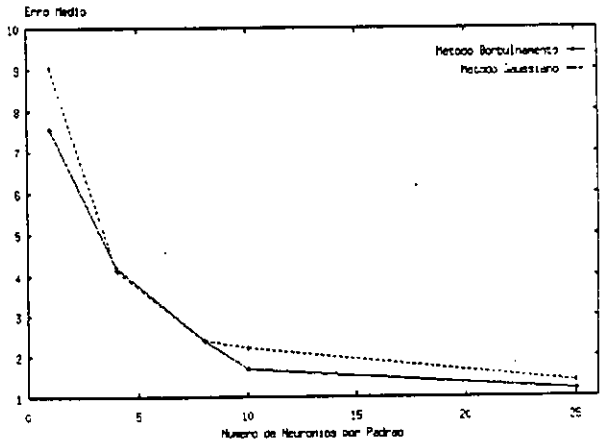


Figura 7: Variação do Erro Médio em Função do Número de Neurônios por Padrão.

A partir desses resultados, verificou-se que o método borbulhamento é eficiente na criação dos clusters, permitindo que a superfície gerada envolva a maioria dos padrões de entrada. O erro médio obtido usando

esse método é relativamente menor que o obtido pelo gaussiano fazendo com que os clusters sejam melhor distribuidos pela rede. Já o método gaussiano espalha mais os neurônios pelo cluster possibilitando uma suavidade na superfície gerada. Assim esses métodos devem ser combinados para, num primeiro intante, distribuir os clusters pela rede e depois ordenar os neuronios dentro de cada cluster (ver Figura 8).

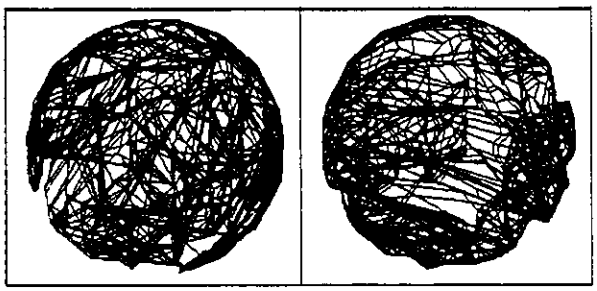


Figura 8: Superfícies Geradas usando os Métodos *Borbulhamento* e *Gaussiano* a partir de uma Esfera.

3.2 Análise do Aspecto Visual da Rede

Esta análise foi feita observando o comportamento dos pesos da rede durante o treinamento. Foi usado como conjunto de treinamento os pontos da superfície de uma *esfera*, de um *cone*, pontos de uma superfície de revolução que se assemelha a um *cálice* e outra a um *chapéu* e pontos da superfície de um *tronco feminino*. Cada experimento foi realizado em duas fase. na primeira partiu-se de uma rede aléatoria e usou-se o método *borbulhamento* para criação dos clusters. Já na segunda fase, utilizou-se a rede obtida na primeira, aplicou-se o método *gaussiano* a fim de suavizar a superfície.

A tabela abaixo mostra os parâmetros usados na primeira faze da modelagem:

Método Borbulhamto					
Exp	**Superf.**	**Rede**	R_i	**NA**	**Taxa**
1	Esfera	38x38	19	50	90%
2	Cone	36x36	18	50	90%
3	Taça	38x38	19	50	90%
4	Chapéu	38x38	19	50	90%
5	Tronco	33x33	25	50	90%

Legendas: Exp = Experimento; Superf. = Superfície Geradora; Rede = Dimensão da Rede; R_i = Raio Inicial da Vizinhança Topológica; NA = Número de Apresentação; Taxa = Taxa de Adaptação

A seguir é apresentada uma tabela[4] com os parametros utilizados na segunda fase dos experimentos:

Método Gaussiano					
Exp	**Superf.**	**Rede**	R_i	**NA**	**Taxa**
1	Esfera	38x38	4	25	40%
2	Cone	36x36	3	25	30%
3	Taça	38x38	4	25	40%
4	Chapéu	38x38	4	25	40%
5	Tronco	33x33	5	25	40%

As figuras seguintes mostram a evolução da modelagem de superfície para os experimentos acima.

A Figura 9 mostra a evolução do experimento 1, onde tem-se a superfície geradora, a rede após o *borbulhamento* e a rede após o método *gaussiano*, repectivamente. Neste experimento foram usados 154 pontos da superfície da esfera como conjunto de padrões.

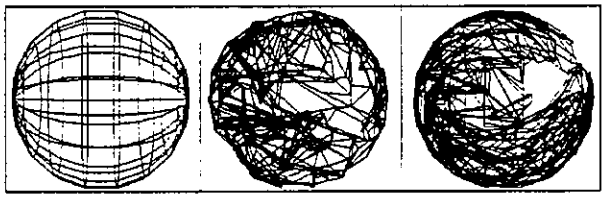


Figura 9: Exp. 1 - Superfície Geradora, Método *Borbulhamento* e Método *Gaussiano*.

Na Figura 10 são mostradas as superfícies obtidas a partir da superfície de um cone usando os dois métodos de treinamento. O conjunto de padrões de treinamento foi composto de 146 pontos.

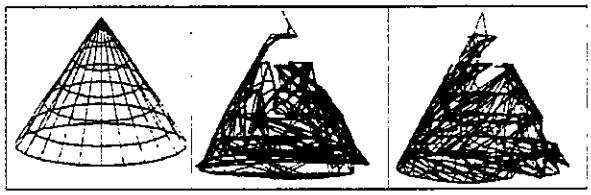


Figura 10: Exp 2 - Superfície Geradora, Método *Borbulhamento* e Método *Gaussiano*.

No experimento 3 foi utilizado, como sólido gerador, uma superfície de revolução semelhante a uma taça . O conjunto de treinamente para esse experimento foi constituído por 160 pontos da superfície desse sólido. A Figura 11 apresenta o comportamento da rede durante esta experiência:

Já no experimento 4, foi gerado uma superfície que se aproxima dos pontos de um sólido de revolução

[4] Usar as legendas da tabela anterior.

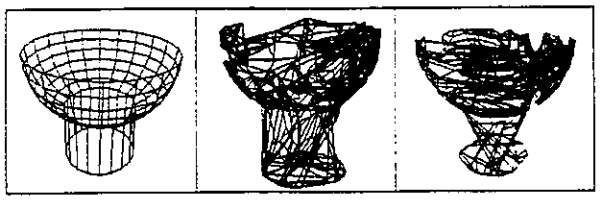


Figura 11: Exp 3 - Superfície Geradora, Método *Borbulhamento* e Método *Gaussiano*.

parecido com um chapéu. O conjunto de treinamento usado continha 61 pontos dessa superfície. A Figura 12 ilustra esse experimento.

Figura 12: Exp 4 - Superfície Geradora, Método *Borbulhamento* e Método *Gaussiano*.

O último experimento usou a escultura de um tronco feminino como superfície geradoura. A Figura 13 mostra a evolução deste experimento. Pode-se perceber o aparecimento de 1 neuronio não comprometido na superfície.

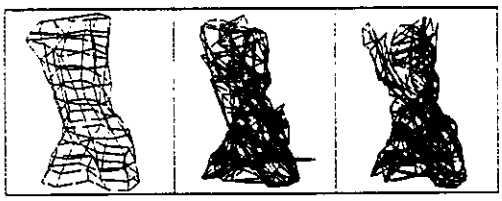


Figura 13: Exp 5 - Superfície Geradora, Método *Borbulhamento* e Método *Gaussiano*.

Através desses experimentos pode-se verificar o resultado das aplicações dos métodos *Borbulhamento* e *Gaussiano* na modelagem de superícies. Variando o número de apresentações, o número de neurônios e sobretudo o raio inicial da vizinhança topológica. pode-se conseguir diferentes graus de suavidade e distribuição de pontos.

4 Conclusões e Trabalhos Futuros

Este trabalho apresentou uma técnica para modelagem de superfície através de Redes Neurais Artificiais. A utilização do modelo neural proposto por Kohonen se mostrou bastante satisfatório, favorecendo a geração e adaptação de superfícies. Este processo auto-adaptativo permitiu a manipulação dessas superfícies gerando novos objetos.

Foram apresentados métodos para distribuição de pontos pela superfície e para aumentar a sua suavidade. No primeiro caso usou-se o *borbulhamento* no processo de treinamento da rede. O controle da suavidade, no entanto, foi obtido utilizando uma *função gaussiana* no algoritmo de aprendizagem.

Uma vez conseguido a modelagem de superfícies. os trabalhos serão voltados para a modelagem de sólidos usando essa mesma abordagem. Isto será feito simulando o método interativo empregado pelo escultor. onde a cada ponto apresentado pelo designer. a rede será adaptada gerando novos objetos. Será usado técnicas de iluminação e texturização para um maior realismo dos sólidos obitidos.

Agradecimentos

Este trabalho contou com o suporte financeiro do CNPq e FACEPE e o apoio técnico do grupo de Redes Neurais dessa Universidade, em particular de Anne M. P. Canuto, Edward Roe e Marcílio C. P. de Souto. Agradecimento especial a Sílvio R. P. C. Melo e a Venusa S. Leitão.

Referências

[1] Marc Berger. *Computer Graphics with Pascal.* The Bejamin/Cummihgs Publishing Company, Inc, University of Colorado, Colorado, 1986.

[2] J. D. Foley and A. Van Dam. *Fundamentals of Interactive Computer Graphics.* Addison-Wesley Systems Programing Series. Addison-Wesley Publishing Company, July 1984.

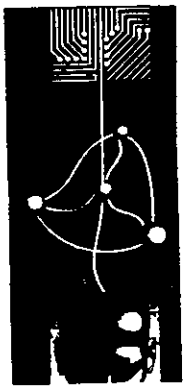
[3] Christoph M. Hoffman. *Geometric & Solid Modeling - an Introduction.* Morgan Kaufman Publishers, Inc., San Mateo, California, 1989.

[4] Jari Kangas, Teuvo Kohonen, and Jorma Laaksonen. *SOM_PAK, The Self-Organizing Map Program Package - Version 1.2.* SOM Program Team of the Helsink University of Technology, Finland, November 1992.

[5] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetcs*, 43:59–69, 1982.

[6] Teuvo Kohonen. *Self-Organization and Associative Memory*, volume 8 of *Springer Series in Informations Sciences.* Berlin - Heidelberg - New York. 3 edition, April 1984.

[7] Yoh-Han Pao. *Adaptive Pattern Recognition and Neural Networks.* Addison-Wiley, 1989.



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

UMA ARQUITETURA PARCIALMENTE RECURSIVA APLICADA A PROBLEMAS DE CLASSIFICAÇÃO

Marcello Baptista de Martino*

Resumo:

As Redes Neurais apresentam-se como uma promissora técnica para aplicações dos Sistemas Elétricos de Potência. Grande parte destas aplicações podem utilizar alguma forma de Classificação em sua resolução. Assim, discutimos e levantamos os requisitos necessários à aplicação de Redes Neurais em Classificação. Apresentamos uma nova arquitetura parcialmente recursiva baseada nos modelos de *Jordan* e *Elman*, selecionamos o algoritmo "Back-Propagation Through Time" para ser utilizado na arquitetura proposta e, finalmente, comprovamos sua eficácia em um exemplo de classificação fornecido por *Telfer*.

I - Introdução

A Inteligência Artificial tem começado a desempenhar um importante papel nos Sistemas Elétricos de Potência, tendo sido utilizada com sucesso em diversas aplicações. Embora a maior parte das aplicações existentes utilize Sistemas Especialistas, certas classes de problemas não têm tido soluções totalmente satisfatórias por causa da inexistência de um modelo matemático adequado (e.g. Controle), da ocorrência de complexas classificações (e.g. Detecção de Falhas) ou do elevado tempo de processamento (e.g. Análise de Contingência). As Redes Neurais devido à sua capacidade de aprendizado através de exemplos, de processamento em tempo real e de generalização, apresentam-se como uma solução promissora a ser explorada.

Os levantamentos realizados sobre as aplicações existentes de técnicas de Inteligência Artificial concluíram que o sucesso destas depende da qualidade do conhecimento adquirido e representado. De um modo geral, as aplicações existentes podem ser divididas em 3 tipos:

- as que buscam otimizar uma ou mais variáveis do sistema (otimização),
- as que relacionam eventos ou estados do sistema à conclusões ou ações sobre este (classificação),
- as que realizam ambas as tarefas.

A Classificação é uma poderosa estratégia de organização do conhecimento que pode ser utilizada em diversas aplicações (e.g. monitoração, predição, interpretação e controle). Assim, o estudo de Redes Neurais em Classificação é fundamental para a resolução de grande parte desses problemas.

* CEPEL - Centro de Pesquisas de Energia Elétrica
Av. Hum, sem número, Cidade Universitária, Rio de Janeiro, RJ, BRASIL, CEP 21.491-590
E-Mail: martino@fund.cepel.br

II - Redes Neurais em Classificação

Uma Rede Neural pode funcionar de dois modos distintos: como um associador de padrões ou como um detector de regularidades. O paradigma Classificador pode ser considerado como um caso particular da associação de padrões, cujo objetivo é aprender a classificar os padrões dos exemplos em categorias distintas, de modo a estar apto a classificar outros padrões quaisquer. Um dos mecanismos mais utilizados neste paradigma é o aprendizado competitivo, no qual os elementos processadores competem entre si para determinar o vencedor. Este paradigma resolve, na verdade, um caso particular da classificação, denominado categorização.

Uma das primeiras formas de representação do conhecimento relativo a problemas de Classificação denomina-se Redes Semânticas, e pode ser modelada por um grafo cujos vértices representam os conceitos envolvidos e cujas arestas representam o relacionamento entre estes conceitos.

Uma Rede Semântica normalmente apresenta arcos de generalização e especialização, organizando os conceitos em uma taxonomia [Woods,91]. A taxonomia pode ser interpretada como uma estrutura para agrupar indivíduos de acordo com características comuns, e a classificação como o processo pelo qual novos conceitos são localizados ou adicionados a uma taxonomia existente.

Tradicionalmente, para resolver problemas de classificação, é criada uma hierarquia de módulos pré-treinados de Redes Neurais para categorização, compondo uma Rede Neural Hierárquica. Tal solução envolve a criação de uma arquitetura diferente para cada problema.

Para que se possa efetivamente utilizar uma única arquitetura de Redes Neurais na resolução de problemas gerais de classificação, deve-se ter o cuidado de analisar as características particulares inerentes a estes problemas.

III - Levantamento dos Requisitos

Primeiramente, é preciso considerar que o especialista, por ser a pessoa mais capacitada na resolução do problema em questão, é quem determina quais são os conceitos significativos e fornece os exemplos adequados. Assim, a rede neural a ser utilizada deve possuir um mecanismo de ***aprendizado supervisionado***.

No paradigma classificador, os padrões de saída são diferentes dos padrões de entrada *(Equação 1)*, indicando a utilização de redes ***hetero-associativas*** em sua resolução.

$$\mathbf{E \Rightarrow S}$$

Equação 1

onde o símbolo "$\Rightarrow$" representa uma dependência conceitual dos conceitos de saída **S** em relação aos conceitos de entrada **E**.

Em problemas de categorização, os conceitos de entrada são totalmente independentes dos conceitos de saída. Porém, em problemas de classificação, um especialista muitas vezes se utiliza de ***conceitos duais*** (conceitos que às vezes servem de entrada e outras, de saída) nos exemplos *(Equação 2)* para sua resolução.

$$\mathbf{(E+D) \Rightarrow (D+S)}$$

Equação 2

onde o símbolo **D** representa os conceitos duais.

Neste tipo de exemplos, os conceitos duais podem ser utilizados tanto como conceitos de entrada quanto como conceitos de saída. Um conceito dual pode, inclusive, ser utilizado para obter outro conceito dual, indicando que a rede deve ser ***recursiva***.

Embora os conceitos duais devam ser utilizados como entrada durante o treinamento, na utilização da rede apenas os conceitos de entrada estarão disponíveis, implicando em ***diferentes conjuntos de treinamento e testes***. A rede

deve ser capaz de obter padrões de saída **S** e padrões duais **D**, a partir de padrões de entrada **E** *(Equação 3)*.

$$E \Rightarrow (D+S)$$

Equação 3

Finalmente, até mesmo os exemplos a serem utilizados durante o treinamento não são "completos", ou seja, o especialista não fornece exemplos em que todas as entradas resultem em todas as saídas. Pelo contrário, ele procura selecionar exemplos gerais e, a partir destes, se especializa em exemplos mais detalhados. Esta caracteristica requer que exista algum mecanismo de ***aprendizado a partir de exemplos de dados incompletos***. Esta não é uma limitação relativa apenas à problemas de classificação, mas comum a todas as aplicações em Redes Neurais.

IV - Arquitetura Proposta

A maioria dos modelos conhecidos possuem aprendizado supervisionado e são hetero-associativas, porém não apresentam uma topologia recursiva. Um exemplo típico é a rede "Back-Propagation", atualmente o modelo mais utilizado em aplicações práticas no mundo.

Os modelos recursivos são pouco utilizados e sua aplicação prática é ainda mais restrita, pois apresentam sérios problemas de convergência e estabilização.

Uma abordagem recente utiliza redes com a maioria das conexões em uma única direção ("feed-foward") e algumas, cuidadosamente selecionadas, na outra direção ("feed-back"), sendo denominadas ***parcialmente recursivas***.

A rede de *Jordan* [Jordan,89] é uma arquitetura parcialmente recursiva *(Figura 1)*, na qual uma camada extra, denominada ***camada de contexto***, copia o padrão de ativação da camada de saída no instante anterior.

A camada de contexto é realimentada por si mesma, de modo a acumular os traços dos valores passados. As conexões recursivas permitem que a camada escondida utilize suas próprias respostas anteriores para orientar seu comportamento subseqüente, criando uma memória na rede. Com uma entrada fixa, sua rede pode ser treinada para gerar uma seqüência de saídas, além de poder ser treinada para reconhecer e distinguir diferentes seqüências de entradas. Uma rede deste tipo é capaz de aprender a emular um Autômato de Estados Finitos.

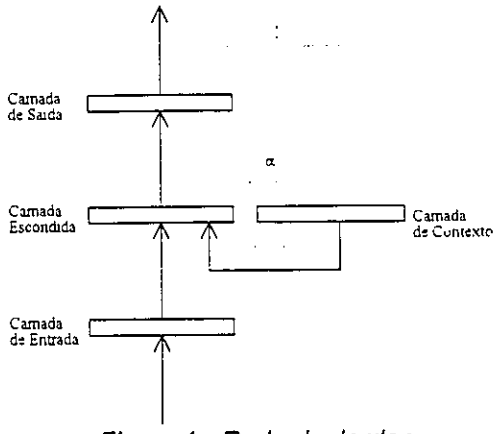


Figura 1 - Rede de Jordan

A rede de *Elman* possui uma arquitetura similar [Elman,90], porém mais simples, na qual a camada de contexto copia os valores da camada escondida ao invés da camada de saída *(Figura 2)*.

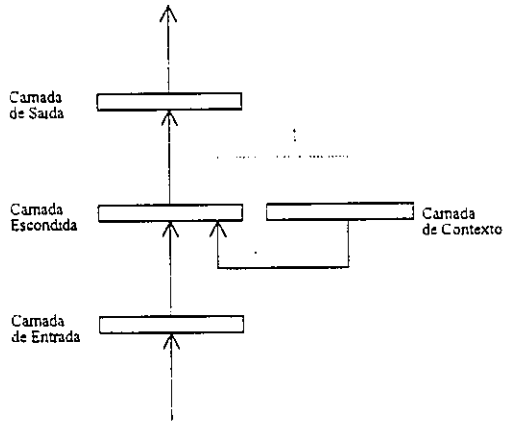


Figura 2 - Rede de Elman

Nesta arquitetura, a camada de contexto armazena o estado anterior da camada escondida, provendo a rede de memória. A camada escondida deve mapear tanto as entradas quanto o estado

anterior nas saídas desejadas. Os padrões internos são salvos como contexto e este, por sua vez, é utilizado na determinação do próximo padrão interno. Deste modo, os padrões internos são sensíveis ao contexto.

Baseados nos modelos de *Jordan* e *Elman*, desenvolvemos uma arquitetura parcialmente recursiva de redes neurais *(Figura 3)* que atende aos requisitos referentes à arquitetura levantados.

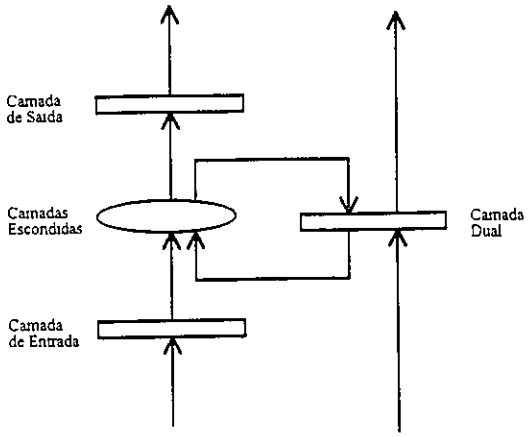


Figura 3 - Arquitetura Proposta

A arquitetura sugerida, além de prover a rede de memória, permite treiná-la com o contexto desejado. Este contexto é aproveitado durante a determinação do próximo contexto, e assim sucessivamente.

V - Seleção do Algoritmo

As redes desenvolvidas por *Jordan* e *Elman* possuem somente conexões recursivas de peso fixo e permitem a utilização do algoritmo de aprendizado ***"Back-Propagation" (BP)***. Distintamente destas, a arquitetura proposta envolve conexões recursivas adaptáveis, não permitindo a utilização deste algoritmo.

O algoritmo "Back-Propagation" pode ser extendido para Redes Neurais arbitrárias [Pineda,89], denominando-se ***"Recurrent Back-Propagation"***, cuja dificuldade é garantir a convergência para estados estáveis, assim como em qualquer rede recursiva. O "Back-Propagation" pode ser considerado um caso particular do "Recurrent Back-Propagation" (RBP).

No mesmo trabalho em que criticou os Perceptrons, *Minsky* mostrou que para qualquer rede neural recursiva existe uma rede "feed-foward" correspondente, com o mesmo funcionamento (em um intervalo finito de tempo). A rede sem realimentação equivalente [Rumelhart,86] pode ser treinada através de uma versão discretizada do "Back-Propagation", conhecido como ***"Back-Propagation Through Time"***.

O algoritmo anterior pode ser adaptado para aprender enquanto os padrões são apresentados, possibilitando sua utilização com seqüências de tamanho indeterminado e em tempo real [Zipser,89], sendo denominado ***"Real Time Recurrent Learning" (RTRL)***. Porém, requer a manutenção, para cada intervalo de tempo, de N^4 derivadas.

Existe, ainda, um outro algoritmo denominado ***"Time-Dependent Recurrent Back-Propagation"***, capaz de treinar uma rede recursiva genérica em intervalos contínuos de tempo [Pearlmutter,89], que pode ser visto tanto como uma extensão do RTRL para intervalos contínuos de tempo quanto como uma extensão do RBP para seqüências dinâmicas.

Como o problema em questão não exige intervalos contínuos de tempo nem aprendizado em tempo real, selecionamos o algoritmo "Back-Propagation Through Time" para ser utilizado na arquitetura sugerida.

Infelizmente, o algoritmo sugerido minimiza o erro entre os padrões obtidos e os desejados, tornando necessário que os dados dos exemplos de treinamento sejam completos.

VI - Back-Propagation Through Time

A rede original é discretizada no tempo através da replicação de cada unidade de processamento em intervalos de tempo. As conexões ligam uma unidade em um instante **t** a outra em um instante **t+1**. A atualização de cada unidade no instante **t+1** é dada por uma função do estado das unidades no instante **t** (e.g. *Equação 4*).

$$U_i(t+1) = g\left(\Sigma w_{ij} U_j(t) + \varepsilon_i(t) \right)$$

Equação 4

De um modo geral, o procedimento deste algoritmo consiste em apresentar o padrão de entrada para o sistema, permitir que ele rode por um determinado número de iterações, comparar o resultado com a saída desejada, retropropagar os erros obtidos pelo mesmo número de iterações e atualizar os pesos.

Assim, uma de suas maiores dificuldades está em determinar um número de iterações que seja suficientemente grande para garantir a resposta desejada e suficientemente pequeno para que não torne o procedimento inviável, como em seqüências compridas ou de tamanho indeterminado. No caso dos problemas de Classificação este número pode ser determinado através da altura da árvore de hierarquia ou da profundidade do grafo correspondente.

Devido à replicação das unidades, o principal problema desse algoritmo é a necessidade de elevados recursos computacionais. Apesar disso, esta restrição se aplica somente à fase de treinamento, pois, uma vez treinada a rede discretizada no tempo, a rede recursiva original com os pesos obtidos pode ser utilizada.

Assim como toda rede recursiva, a arquitetura proposta pode ser igualmente discretizada no tempo *(Figura 4)*.

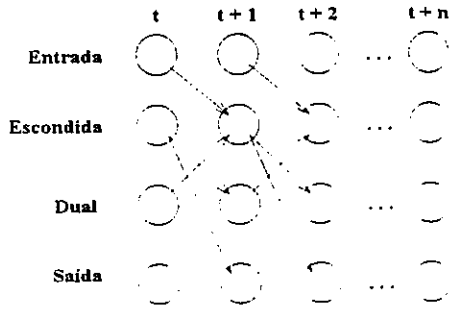


Figura 4 - Discretização da Arquitetura

É possível demonstrar que, independentemente do problema, a arquitetura pode ser discretizada em apenas dois intervalos de tempo.

VII - Testes

Como teste utilizamos uma ligeira variação do exemplo de um artigo [Telfer,91] publicado no periódico *Neural Networks*, em que especialistas levantaram uma classificação *(Figura 5)* para determinação de diferentes aeronaves.

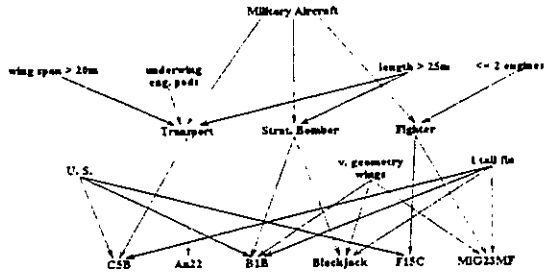


Figura 5 - Exemplo (Telfer,91)

No exemplo, foram utilizados conceitos duais (Transporte, Bombardeiro e Caça) a partir dos conceitos de entrada para determinar os conceitos de saída (C5B, An22, B1B, BlackJack, F15C e Mig23MF).

Para implementação deste exemplo na arquitetura proposta, associamos cada conceito a um elemento processador. Cada elemento das camadas de entrada, dual e saída representa, respectivamente, um conceito de entrada, dual e de saída. Assim, a rede neural construída tem 8 elementos de entrada, 3 duais e 6 de saída. Além destes, foram utilizados 4 elementos na camada escondida.

Para treinar uma rede neural normal (e.g. "Back-Propagation") é preciso levantar os exemplos completos do mapeamento entre os conceitos de entrada e saída. Para treinar a arquitetura proposta é necessário incluir também os conceitos duais nos exemplos, que podem ser obtidos diretamente da hierarquia de conceitos *(Figura 5)*.

	1	2	3	4	5	6	7	8	9
Militar	1	1	1	1	1	1	1	1	1
Envergadura>20m	1	1	1	1	0	0	1	0	0
Tanque nas Asas	1	1	0	0	0	0	1	1	0
Comprimento>25m	1	1	1	1	0	0	1	1	0
Motores<3	0	0	0	0	1	1	0	0	1
Americano	1	0	1	0	1	0	0.3	0.3	0.3
Asas em V	0	0	1	1	0	1	0.3	0.3	0.3
Quilhas=1	1	0	1	1	0	1	0.3	0.3	0.3
Transporte	1	1	0	0	0	0	1	0	0
Bombardeiro	0	0	1	1	0	0	0	1	0
Caça	0	0	0	0	1	1	0	0	1
C5B	1	0	0	0	0	0	0.5	0	0
An22	0	1	0	0	0	0	0	0	0
B1B	0	0	1	0	0	0	0	0	0
BlackJack	0	0	0	1	0	0	0	0.5	0
F15C	0	0	0	0	1	0	0	0	0.5
Mig23MF	0	0	0	0	0	1	0	0	0.5

Tabela 1 - Treinamento e Testes

Para ensinar a rede foram utilizados os seis exemplos *(Tabela 1: exs.1 a 6)* fornecidos (um para cada aeronave). Para testá-la foram utilizados três outros exemplos *(Tabela 1: exs.7 a 9)* que servem para determinar se a rede aprendeu corretamente a hierarquia desejada. Nestes exemplos, os três conceitos de entrada que não foram utilizados pelos especialistas (Americano, AsasEmV, Quilhas=1) na determinação dos conceitos duais são representados por um valor médio (0.3). Os testes mostram que a rede foi capaz de determinar com precisão (valor 1) o tipo (Transporte, Bombardeiro ou Caça) e de sugerir uma aeronave (valor 0.5).

O simulador utilizado foi o SNNSv3.1 ("Stuttgart Neural Network Simulator"), da Universidade de Stuttgart, em uma "workstation" da DEC. Devido à implementação desse algoritmo neste simulador, foram necessárias 6 iterações ao invés de 3 (altura da hierarquia) para cada exemplo. Também foram realizados testes com sucesso na Detecção de Falhas em Transformadores. Os resultados destes testes serão em breve divulgados em artigos mais específicos.

VIII - Conclusões

A arquitetura proposta pode ser interpretada de dois modos distintos: como uma rede parcialmente recursiva com memória determinada externamente, ou como uma arquitetura única equivalente a redes neurais "feed-foward" hierárquicas.

O desenvolvimento de um algoritmo para aprendizado a partir de exemplos com dados incompletos representaria um avanço significativo na aplicabilidade das Redes Neurais, e permitiria a plena utilização da arquitetura proposta.

Um dos resultados obtidos mais interessantes é que o maior problema do algoritmo selecionado (a determinação do número necessário e suficiente de intervalos de tempo) desaparece quando da aplicação deste na arquitetura proposta.

Finalmente, os testes efetuados demonstraram a eficácia da utilização da arquitetura em problemas de classificação, e demonstraram seu potencial para utilização em aplicações ainda mais complexas (e.g. Controle).

IX - Bibliografia:

[Elman,90] "Finding Structure in Time", J.L.Elman, Cognitive Science, vol.14, no.2, pp.179-211

[Jordan,89] "Generic Constraints on Underspecified Target Trajectories", M.I.Jordan, IJCNN'89, vol.1, pp.217-225

[Pearlmutter,89] "Learning State Space Trajectories in Recurrent Neural Networks", B.A.Pearlmutter, ICNN' 89, vol.II, pp.365-372

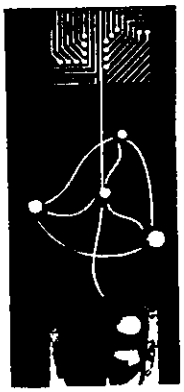
[Pineda,89] "Recurrent Back-Propagation and the Dynamical Approach to Adaptive Neural Computation", F.J.Pineda, Neural Computation, vol.1, pp.161-172

[Rumelhart,86] "Parallel Distributed Processing: Explorations in the Microstructure of Cognition", D.E.Rumelhart & J.L.McClelland, The MIT Press, Cambridge, Massachusetts

[Telfer,91] "Neural Closure Associative Processor", B.Telfer & D.Casasent, Neural Networks, vol.4, pp.589-598

[Woods,91] "Understanding Subsumption and Taxonomy: A Framework for Progress", W.A.Woods, "Principles of Semantic Networks: Explorations in the Representation of Knowledge", J.F.Sowa, M.Kaufmann Publishers Inc.

[Zipser,89] "Encoding Sequential Structure: Experience with the Real-Time Recurrent Learning Algorithm", D.Zipser & A.W.Smith, IEEE ICNN'89, vol.I, pp.645-648



1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajubá
Itajubá, 24 a 27 de outubro de 1994

I-DYNA: An Architecture for Integrating Reacting, Planning, Learning and Self-Awareness in Autonomous Agents

Camelia Florela VOINEA
Universita' degli Studi di Torino, Italia
Dipartimento di Informatica, cso.Svizzera 185, 10149 Torino
email: camelia@di.unito.it

This work represents a connectionist approach to adaptive reasoning. It proposes a new version of the Dyna class of architectures, named Introspective-Dyna, which is aimed at integrating reacting, planning and learning with self-awareness of an agent in making an action decision. The introspective component of I-Dyna represents for the agent a belief subsystem able to help the agent having an attitude which emerges from within its internal representation of the world. An introspective prediction process is aimed at replacing a belief inferential process. Appropriateness (as soundness) and usefulness (as completeness) of taking an action are judged by means of faithfulness and fulfilment factors, as introspective explanative components of current decision making step. A double-sided temporal difference method is underlying each prediction step: the agent must look-ahead one step and choose an action with most-reachable anticipated rewards (real prediction) and must look backward (at least) one step and choose the action with most-acceptable introspectively anticipated results (introspective prediction). Two version of the I-Dyna algorithm, synchronous and asynchronous, are presented and discussed.

Keywords: reinforcement learning, adaptive reasoning, adaptive neural network, autonomous agents.

1. Introduction

The paradigm of learning from reinforcement is that of the interaction between an agent and its environment in which the agent controls the environment by continuously interpreting at the sensorial level the consequences of its own actions. The agent permanently adapts its reaction by selecting appropriate actions in a sequence which depends on a goal, on time limits and on certain bounds and constraints regarding the physical resources of the environment and of the agent itself. The main theoretical basis of this approach is provided by researches in the area of neural modelling of perception and planning/learning capabilities of adaptive autonomous agents.*Dynamic programming* [Howard, 1960], [Bellman, Dreyfus, 1965] provides an useful but poor model of learning. One of the most important limitations of the dynamic programming is that an action should be tried in order to observe the reward. In the typical case, the reward is associated with the completion of a goal, therefore it could only become available at the end of a sequence of actions. In a sense, dynamic programming methods work backwards from the end of a decision task to its beginning, calculating information pertinent to decision making at each stage based on information previously calculated from that stage to the task's end. As a result of this back-to-front processing, it is difficult to see how dynamic programming can be related to learning processes that operate in real-time as an agent interacts with its environment.*Temporal Differences methods* [Sutton, 1988] can accomplish much the same result, through repeated trials, instead of explicit back-to-front computation. TD methods represent a class of incremental learning procedures specialized for prediction problems. Whereas conventional prediction-learning methods are driven by the error between the real output and the desired one, the temporal difference methods are driven by the error between temporally succesive predictions: with them , learning occurs whenever there is a change in prediction over time.The TD class of prediction methods relies on the simple principle that the observable effects of a sequence of events/actions may be interpretated as ways in which the fundamental laws actually governing the relationships between sequences of events and sequences of observable effects manifest themselves. This principle has been widely applied in pattern recognition and classification. Methods of intelligent control are also using these kind of approach in trying to combine active control with on-going exploration of the behavior of the system to be controled. Combined with a simultaneously formation and adaptation of a world model underlying the prediction process, these methods could provide an appropriate framework for studying the prediction as well as explanative capabilities of embedded systems.*Dyna* [Sutton, 1990] represents a class of simple architectures integrating planning, reacting and learning. Intuitively, Dyna is based on the idea that planning is like trial-and-error learning from hypothetical experience. The Dyna theory is based on the theory of Dynamic Programming, to temporal-difference learning and to AI methods for planning and search. For a fixed policy, Dyna-PI is simply a reactive system, but its policy is continually adjusted by an integrated planning and learning process. The policy is viewed as a plan which is completely conditioned by the current input. The planning process, also called relaxation planning, is incremental and consists of shallow searches, each typically of one step ply, ultimately producing the same result as an arbitrarily deep conventional search. The decision making process is based on continually adjusting the evaluation function in such a way that credit is propagated to the appropriate steps within action sequences.The same algorithm is applied both to real experience (resulting in learning) and to hypothetical experience generated by the world model (resulting in relaxation planning). The results in both cases are accumulated in the policy

and the evaluation function. In spite of its elegance and simplicity, there are however, some weaknesses in the Dyna architecture.The fundamental achievement of Dyna resides, undoubtedly, in the existence of the hypothetical experience itself, in the basic idea of relaxation planning: the use of a "minimum" but, in the same time, "effective" plan about how to act in the next step, the accumulation of both planning and acting experience in the on-going policy and evaluation function with evident improvement of acting process itself, the simultaneity of reacting and planning and the use of the same source of information for updating both reaction and planning in an incremental way. The main weakness of Dyna is that the hypothetical experience does not result also in insight or cognitive capabilities; the agent does not know about the world and about himself more than his evaluation function does, that is, an external measure of his accomplishing a given goal. If this external measure changes, the knowledge the agent has about the world becomes again of no significance. The hypothetical experience in Dyna is meant to express the reasoning of the agent and it has a cognitive aspect inasmuch the plan itself expresses the knowledge the agent has about the world in the current situation, and which is evident by means of the evaluation function. Nevertheless, the agent needs to have an elementary condition of intelligence: self-awareness. How can the agent acquire or be provided with this capability actually represents the aim of this approach.

2. I-Dyna Assumptions

The rationale for these assumptions is that the use of hypothetical experience in prediction requires:(1) a model of the world, particularly a model which is actually formed and updated while the agent interacts with the process to be controlled; the problem is that this kind of model has a high degree of uncertainty in representing the world since it is formed while the world itself is being explored;(2) a theoretical separation, between the input/output provided by the real process and what could represent an input/output for the planning task. Existing approaches on combining reacting and planning in reinforcement learning, especially Dyna [Sutton, 1990], planning with time constraints [Kaelbling, 1991], planning for closed-loop execution using partially observable Markovian decision processes [Chrisman, 1992], have investigated the problem of performing planning in an incremental manner simultaneously with interacting with the process to be controlled. Nevertheless, neither of these approaches suggests an epistemic view of learning by reinforcement, they concentrate mainly on the idea of learning as an incremental task typically as a trial-and-error process and on the optimality principle as a substitute for a more complex belief subsystem able to provide the controller with the capability of interpreting the cognitive value of reinforcement and to have an epistemic attitude with respect to it.

The main working assumptions in this approach are the followings:

1- the choice of an action is influenced by the feed-back provided by the controlled process itself, therefore we will speak about a **closed-loop control architecture**;

2- the input of the decision making process and the internal state have separate representations, since the state cannot be completely observed , so that we will speak about **partially observable Markov processes**;

3- the agent embodies a model of the world and is provided with the capability of reflecting upon the choice of his actions and the effects of his acting by means of an **introspective belief subsystem** which is supposed to express his attitudes towards his own reasoning capabilities.

3. Model Formation and Adaptation in I-Dyna

The particular way in which the present approach makes reference to the idea of insight capabilities an agent could have starts from the idea of "subjective rewards " [Watkins, 1989], and assumes that the action decision making in embedded systems is obviously determined both by the external world and by the internal representation the agent has of his world. The idea of primitive learning [Watkins, 1989] is that an agent could learn from trial-and-error on a stimulus-response base by optimizing his choices with respect to the resources of the environment and to his own resources. Nevertheless, if we want this agent to behave intelligently, we have to accept the idea that it builds a model of its world while interacting with it and, moreover, it has introspective capabilities, that is , it is able to reflect upon the workings of its own cognitive functions.

This approach makes reference to the concepts introduced, from one side, by the epistemologic approach of Gardenfors concerning the dynamics of epistemic states, [Gardenfores, 1988] and, from the other side, by the explanative introspective approch introduced by Konolige [Konolige, 1985]. Upon making the decision on which action to take next, the agent is supposed to make a prediction on the basis of the interpretation he is able to give to the information at hand. This interpretation expresses the knowledge he has in that moment about the state in which he find himself w.r.t the environment and the attitude he takes w.r.t the possible results of taking an action in the given state. This attitude could be expressed also in terms of a prediction in which the subjective rewards guiding the evaluation function are the faithfulness and the fulfillment [Konolige, 1985] factors. This kind of prediction will be further denoted in this paper as *introspective prediction.* The notions of *faithfulness* and *fulfillment* are meant to express here the degrees of epistemic entrenchment [Gardenfores, 1988]. Intuitively, we could imagine the human decision making like a double-linked relationship (see figure 1) between what his goal is and what he believes he can do to achieve this goal:

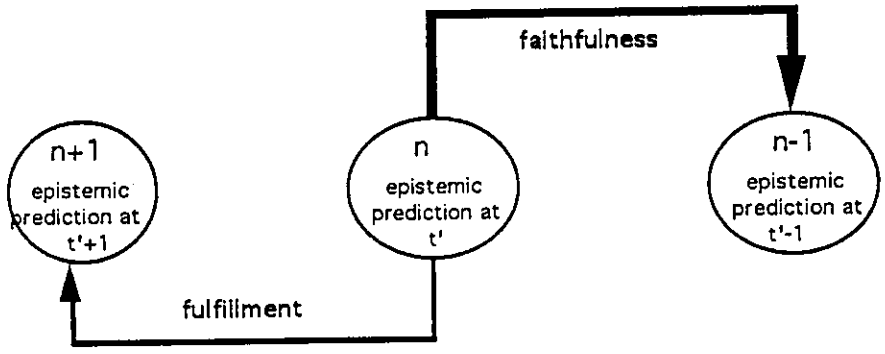


Fig.1. The double-sided temporal difference method: the agent looks-ahead one step to find the most-reachable (really anticipated) action and looks-backward (at least) one step to find the most-acceptable (introspectively posticipated) action.

The *introspective prediction* is a notion denoting the process of recursively generation of the implicit beliefs simultaneously allowed and required by the real prediction process itself. We define this notion for its particular use for understanding the meaning of what precisely I-Dyna architecture embodies as hypothetical experience. The hypothetical experience is a notion used in I-Dyna to denote the introspective beliefs and it is assimilated to some kind of an epistemic commitment function [Gardenfores, 1988]. The input of the introspective module (IP) is represented by a query from P, actually from the real decision making process, addressed to IP.

The output of IP module is an epistemic attitude w.r.t the real prediction performed by the P module (see figure 2). The fundamental relationship between temporarily succesive total predictions (both real and introspective) is given by :

$$E_t \leftarrow E_{t-1} + B^{n+1}$$

where E represents the real prediction at two successive real time moments t and t-1, whereas B represents the introspective prediction at the introspective level (epistemic state) n+1 and at a hypothetical moment of time which will be further denoted by t'. The real prediction E at the current moment t depends on the previous prediction E at t-1 and on what can be derived at a successive introspective level (n+1) if starting from the beliefs at the current introspective level (n) and evaluating the acceptability of what is derived [B at the level n+1] with respect to its base [B at the level n].

The connectionist representation of the introspective prediction allows us to assimilate the degree of epistemic entrenchment [Gardenfors, 1988] to the weights between succesive levels of introspective beliefs (faithfullness and fulfillment factors). The structure of the network itself will provide us with the epistemic commitment function and, moreover, with an explanation of taking one epistemic attitude or another.

The architecture this approach is relying on, Introspective-Dyna, essentially expresses the following idea: the choice of an action in making a decision is the result of combining information provided by the real process itself , I(t), (i.e.: the controlled process, assuming that its outputs are measurable), the agent's knowledge about the process to be controlled, E(t), (expressed in terms of the predictions the agent is able to make upon taking an action in a given state) and the agent's beliefs expressed as introspective predictions, B(n,t'), (i.e: predictions made only on the basis of the previous beliefs about the effects of taking an action in a given situation).

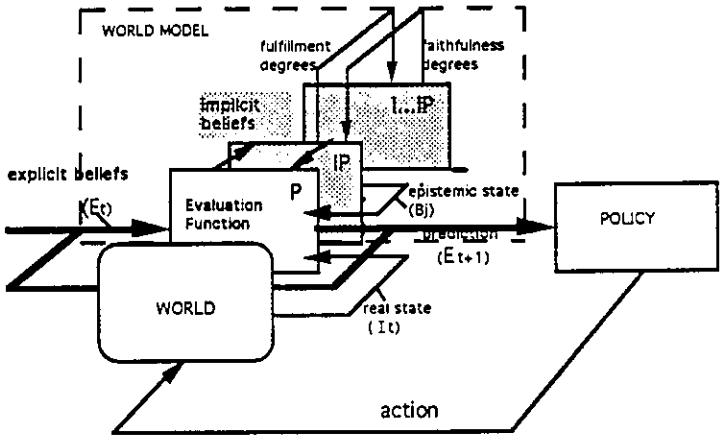


Fig.2. The I-Dyna architecture. Making an acting decision relies on both the world model and the real world. The evaluation function receives both real state information and epistemic state information.

This approach relies, in essence, on the idea that any acting decision an agent could ever make basically expresses his beliefs about his interacting with the world. These beliefs are obviously generated by the model the agent has of its world and by the information (continuously) provided by the world. In making a decision, the agent actually transforms this information into knowledge about the world and about the possible future results (desirable or undesirable) of its interacting with the world.

4. Algorithms

Synchronous I-Dyna Algorithm : *e is the variable for the total prediction and it represents the output, i.e. the action to be taken;ba and breq are the variables for the introspective prediction (the "answer" and the "request", respectively); p is the variable for the real prediction, x,y are real states, r is the reward received from the real process and γ is the discounting factor; n is the variable for the epistemic state, a is the variable for faithfulness factor ;and t is the real time;* initially, e and b coincide; the real time, t, and the introspective time, t', are considered equivalent and simultaneous; each network (RNN - real network, INN - introspective network) is updated by using the double-sided temporal difference method as follows:

for each prediction step: take the action **a** in the real state **x** and introspective state **n**:

update_RNN: $b_{req}(n) \leftarrow i(t) + [p_t(x) + b_a(n-1) + c]$, or

$b_{req}(n) \leftarrow i(t) + e(t)$

$p_t(x) \leftarrow r + \gamma * p_{t-1}(y)$

update_INN: $b_a(n) \leftarrow w_{bb}(n) * b_a(n+1)$

evaluation_function: $e(t+1) \leftarrow e(t) + b_a(n)$

Asynchronous I-Dyna Algorithm: *e is the variable for the total prediction and it represents the output, i.e. the action to be taken;ba and breq are the variables for the introspective prediction (the "answer" and the "request", respectively); p is the variable for the real prediction, x,y are real states, r is the reward received from the real process and γ is the discounting factor; a is the variable for faithfulness factor ;n is the variable for the epistemic state; t is the real time; t' is the introspective time;*

initially, e and b coincide; the real time, t, and the introspective time, t', are not anymore considered equivalent or simultaneous; the relationship between real time and introspective (hypothetical) time is intuitively presented in fig.6 [for each real time prediction (real time unit), the algorithm performs an intrspective prediction over a sequence of n epistemic states, for each state sequence using a corresponding introspective time units; there is, therefore, a 3-level prediction, from which only the first is visible)]; each network (RNN - real network, INN - introspective network, TNN - tendency network) is updated by using the double-sided temporal difference method as follows:

MAKE_REQUEST:

update_RNN: $b_{req}(t,n) \leftarrow i(t) + b_a(n-1,t') + c$

$p_t(x) \leftarrow r + \gamma * p_{t-1}(y)$

MAKE_ANSWER:

update_INN: $b_a(n,t') \leftarrow a(n,t') * b_a(t'+k,n+j)$

GET_ANSWER (PLAN_DELIVERANCE):

if u lower than *Bounds*, then , for some k and some j, such that :

update_TNN: $b_a(n,t') \leftarrow a(n,t') * b_a(t'+k, n+j)$

evaluation_function: $e(t+1) \leftarrow p(t)$ if $t' \ll t$

$e(t+1) \leftarrow p(t) + b_a(n)$ if $t' \equiv t$

and

$n+1 \leftarrow n+j$

5. Conclusions and Future Work

5. Conclusions and Future Work

1. I-Dyna suggests the replacement of an unique external measure of success in decision making with a composed measure of success: internal and external. However, an internal measure of success is bound on the idea of an *internal model of the world*, which comes to the point of "subjective rewards" introduced by Watkins.Trying to overcome the problem of identifying a "content" to these "subjective rewards" scales up to another problem: how does an agent succeed to know its own environment?, what is the precise content of a "cognitive state"?, how does the agent pass from one cognitive state to another ?, and so forth. Thus, the hypothetical experience and relaxation planning in Dyna have inspired another version of Dyna itself: I-Dyna, which is an architecture for integrating reacting, planning and learning with the self-awareness of an embedded system.
2. The idea of relaxation planning has been enhanced in I-Dyna by two types of prediction: prediction of the sensors-measurable future results of taking an action and prediction of the future cognitive attitude (acceptance, refusal) towards the sensors-provided results. It is introduced the notion of *introspective prediction* and its role in substituting a belief inferential process with an introspective prediction process. An *acceptability-function* is therefore recursively constructed from both real data and introspective knowledge.
3. The only prediction rule used here is a temporal difference rule: in order to predict the results of taking an action, the agent must look-ahead (at least) one step and choose the action with most-reachable (anticipated) results, but in order to predict his attitude towards these results, the agent must look-backward (at least) one step and choose the action with most-acceptable (introspectively posticipated) results. We call this *double-sided temporal difference method* .
4. The intention of this kind of approach is to provide the agent with the capability of react and reflect upon its own acting, finding therefore *explanations* for its actions in terms of both the real data and of the model of the environment in which it finds itself.
5. Preliminary results are provided by a *connectionist implementation* of I-Dyna.

References

[**Anderson, 1987**] Anderson, C., W., "Strategy Learning With Multilayer Connectionist Representations", in Proceedings of the 4th International Workshop on Machine Learning, Irvine, California, Morgan and Kaufmann , 1987.

[**Barto, Sutton, Watkins, 1989**] Barto, A., Sutton, R.S., Watkins, C.J.C.H., "Learning and Sequential Decision Making", COINS Technical Report 89-95, September 1989.

[**Barto, Bradtke, Singh, 1991**] Barto, A., Bradtke, S.J., Singh, S.P., "Real Time Learning and Control Using Asynchronous Dynamic Programming", Technical Report 91-57, Amherst, MA: University of Massachusetts, COINS Dept, 1991.

[**Bellman, Dreyfus, 1965**] Bellman, R.E., Dreyfus, S.E., "La programmation dynamique et ses applications", Dunod, Paris, 1965.

[**Chrisman, 1992**], Chrisman, L., "Planning for Closed-Loop Execution using Partially Observable Markovian Decision Processes, from AAAI Spring Symposium Series: Control of Selective Perception, Sytanford, Univ., March, 1992.

[**Gardenfors, 1988**] Gardenfors, P., "Knowledge in Flux. Dynamics of Epistemic States", M.I.T. Press, 1988.

[**Howard, 1960**] Howard, R.A., "Dynamic Programming and Markov Processes", M.I.T. Press, 1960.

[**Kaelbling, 1990**] Kaelbling, L.Pack, "Learning in Embedded Systems", Ph.D. Thesis, Stanford University, U.S.A., 1990.

[**Konolige, 1985**] Konolige, K., "A Computational Theory of Belief Introspection" in Proceedings of the 9th International Joint Conference on Artificial Intelligence, 18-23 August, 1985, Los Angeles, California, A.Joshi (Ed.), Morgan Kaufmann Publishers..

[**Millan, Torras, 1992**] Millan, J. del R., Torras, C., "A Reinforcement Connectionist Approach to Robot Path Finding in Non-Maze-Like Environments", in Machine Learning Vol. 8, No. 3-4, May 1992, Kluwer Academic Publishers.

[**Rabiner, 1989**] Rabiner, L.R., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", Proceedings of the IEEE, Vol.77, No.2, February 1989.

[**Singh, 1992**] Singh, S.P., "Solving Multiple Sequential Tasks Using a Hierarchy of Variable Temporal Resolution Models", Machine Learning Conference, 1992.

[**Sutton, Barto, 1981**] Sutton, R.S., Barto, A., "An Adaptive Network That Constructs and Uses an Internal Model of Its World", in Cognition and Brain Theory, Vol. IV, No.3, 1981, Lawrence Erlbaum Associates Publishers, Hillsdale, New Jersey.

[**Sutton, Pinette, 1985**] Sutton, R.S., Pinette, B., "The Learning of World Models By Connectionist Networks", in Proceedings of the 7th Annual Conference of the Cognitive Sciences Soc., 1985.

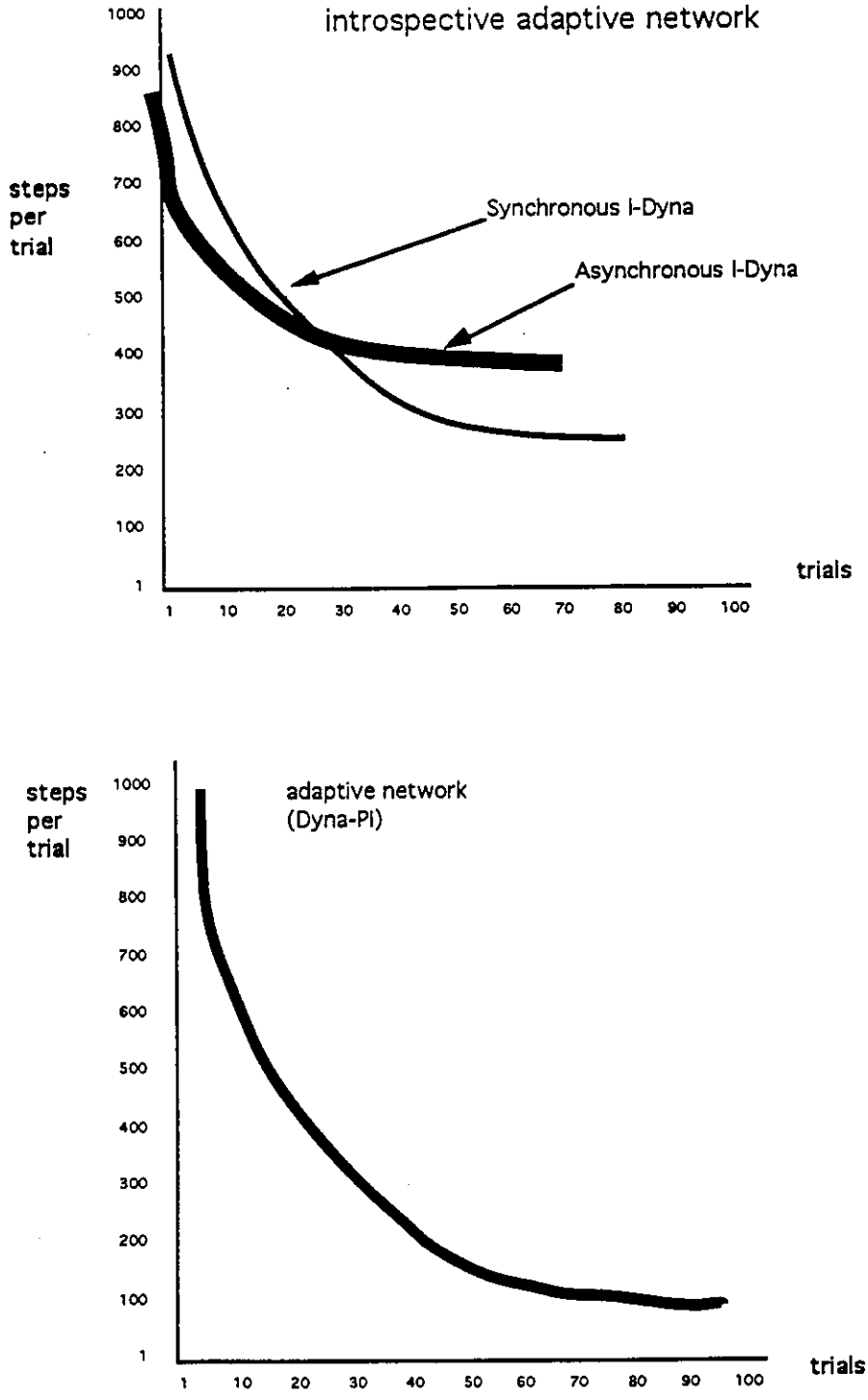
[**Sutton, 1988**] Sutton, R.S., "Learning to Predict by the Methods of Temporal Differences", in Machine Learning No.3: 9-44, Kluwer Academic Publishers, Boston, 1988.

[**Sutton, 1990**] Sutton, R.S., "Integrated Architectures for Learning, Planning and Reacting Based on Approximating Dynamic Programming", in Proceedings of the 7th International Conference on Machine Learning, June, 1990.

[**Sutton, 1991**] Sutton, R.S., "Planning by Incremental Dynamic Programming", in Proceedings of the 9th International Workshop on Machine Learning, 1991, Morgan and Kaufmann.

[**Voinea, 1994**], Voinea, C.F., "Model Formation and adaptation in Reinforcament Learning", in Proceedings of the International Conference on Neural Modeling, University of Lyon, Lyon, France (to appear).

[**Watkins, 1989**] Watkins, C.J.C.H., "Learning from Delayed Rewards", Ph.D. Thesis, King's College, 1989.



Learning Performances of the Introspective Adaptive Network.

Índice por Autor

Apoio:

CEMIG

FUPAI

helibras

BRAHMA Nº1

FAPEMIG

GEC ALSTHOM T&D

American Airlines

CNPq

FAPERJ

COAPI

IBM Brasil

Este trabalho foi impresso e montado com a colaboração da Companhia Energética de Minas Gerais - CEMIG