15th Triennial World Congress, Barcelona, Spain

TRAINING NEURAL NETWORKS AND NEURO-FUZZY SYSTEMS: A UNIFIED VIEW

António E. Ruano, Pedro M. Ferreira, C. Cabrita, S. Matos

ADEEC and Institute of Systems & Robotics,
Faculty of Science & Technology, University of Algarve, 8000 Faro Portugal

Abstract: Neural and neuro-fuzzy models are powerful nonlinear modelling tools. Different structures, with different properties, are widely used to capture static or dynamical nonlinear mappings. Static (non-recurrent) models share a common structure: a nonlinear stage, followed by a linear mapping. In this paper, the separability of linear and nonlinear parameters is exploited for completely supervised training algorithms. Examples of this unified view are presented, involving multilayer perceptrons, radial basis functions, wavelet networks, B-splines, Mamdani and TSK fuzzy systems. *Copyright © 2002 IFAC.*

Keywords: Neural Networks; Neuro-Fuzzy Systems; Multilayer Perceptrons; Radial Basis Functions; Wavelet Neural Networks

1. INTRODUCTION

During the recent years, feedforward neural networks and neuro-fuzzy systems have been increasingly recognized as powerful nonlinear black-box modelling tools, whether for static or dynamic systems. Several authors (see, for instance, Sjoberg *et al.*, 1995) recognized that although different neural networks or neuro-fuzzy systems are employed, they share a common structure: they can be envisaged as a two-stages model, the first performing a nonlinear mapping from an input space to an intermediate space (it will be denoted here as a basis function space), usually of greater dimensionality, and a latter stage, consisting of a linear mapping between the basis function space and the output space.

What is not common to find in the literature is the explicit exploitation, for training purposes, of this common nonlinear-linear topology. This is highlighted in the present paper, where different types of neural and neuro-fuzzy structures will be considered. The next section will introduce a reformulated criterion, different supervised training methods, and how to compute the needed derivatives for each model type. Multilayer perceptrons, radial basis functions, wavelet networks, B-splines, Mamdani and TSK fuzzy systems are considered here. Section 3 shows examples of the application of this reformulated criterion for different examples, and for each model type. Conclusions are drawn in Section 4.

2. A NEW TRAINING CRITERION

The aim of training the different models is to find the values of their parameters that minimize the sum of the square of the errors between the target and the actual output:

$$E = \frac{1}{2}\boldsymbol{e}^T\boldsymbol{e} = \frac{1}{2}\sum_{i=1}^{N} e^2[i] \quad , \tag{1}$$

where $e[i] = o(w)[i] - t[i]$, o being the model output, t the desired output and w the model parameters. Without lack of generality we shall consider in this paper only one output, i.e., we are considering a MISO model.

As the models considered have a layered structure (q being the number of layers), let us partition the parameter vector as:

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}^{(q-1)} \\ \begin{bmatrix} \boldsymbol{w}^{(q-2)} \\ \cdots \\ \boldsymbol{w}^{(1)} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix} \tag{2}$$

where $\boldsymbol{w}^{(i)}$ denotes the matrix of weights connecting layer i to layer i+1, in vectorial form, $\boldsymbol{u}$ denotes the (linear) weights that connect to the output neuron, and $\boldsymbol{v}$ denotes all the other (nonlinear) weights. Using this convention, and as in all models the output layer is linear, the output vector can be given as:

$$\boldsymbol{o}^{(q)} = \left[\boldsymbol{O}^{(q-1)} \; 1\right]\boldsymbol{u} \tag{3}$$

In (3), a bias is assumed to be employed in the output neuron. If this is not the case, then the column of ones in (3) should be omitted. When (3) is substituted in (1), the resulting criterion is:

$$\Omega(\boldsymbol{u}, \boldsymbol{v}) = \frac{\left\| \boldsymbol{t} - \left[\boldsymbol{O}^{(q-1)} \; 1\right]\boldsymbol{u} \right\|_2^2}{2} \tag{4}$$

The dependence of this criterion on $\boldsymbol{v}$ is nonlinear and appears only through $\boldsymbol{O}^{(q-1)}$; on the other hand, Ω is linear in the variables $\boldsymbol{u}$, i.e. the weights in criterion (4) can be separated into two classes: nonlinear ($\boldsymbol{v}$) and linear ($\boldsymbol{u}$) weights. For any value of $\boldsymbol{v}$, the minimum of (4) w.r.t. $\boldsymbol{u}$ can be found using a standard least squares solution. The optimum value of the linear parameters is therefore conditional upon the value taken by the nonlinear variables.

$$\hat{\boldsymbol{u}}(\boldsymbol{v}) = \left(\left[\boldsymbol{O}^{(q-1)}(\boldsymbol{v}) \; 1\right]\right)^{+}\boldsymbol{t} \tag{5}$$

In (5), a pseudo-inverse is used for the sake of simplicity; however, it should be noticed that $\left[\boldsymbol{O}^{(q-1)} \; 1\right]$ is assumed to be full-column rank. Denoting this matrix by $\boldsymbol{A}$, for simplicity, the last equation can then be replaced into (4), creating therefore a new criterion:

$$\psi(\boldsymbol{v}) = \frac{\left\| \boldsymbol{t} - \boldsymbol{A}\boldsymbol{A}^{+}\boldsymbol{t} \right\|_2^2}{2} = \frac{\left\| \boldsymbol{P}_{\boldsymbol{A}_\perp}\boldsymbol{t} \right\|_2^2}{2}, \tag{6}$$

where the dependence of $\boldsymbol{A}$ on $\boldsymbol{v}$ has been omitted. In (6) $\boldsymbol{P}_{\boldsymbol{A}_\perp}$ is the orthogonal projection matrix to the complementary space spanned by the columns of $\boldsymbol{A}$.

This new criterion (6) depends only on the nonlinear weights, and although different from the standard criterion (4), their minima are the same.

We can therefore, instead of determining the optimum of (4), first minimize (6), and then, using $\hat{\boldsymbol{v}}$ in eq. (5), obtain the complete optimal weight vector $\hat{\boldsymbol{w}}$.

This new criterion reformulates the training problem in its correct formulation: an iterative algorithm should look for the best nonlinear mapping that produces the best basis functions for the particular problem. Besides reducing the dimensionality of the problem, the main advantage of using this new criterion is a faster convergence of the training algorithm, a property which seems to be independent of the actual training method employed. This reduction in the number of iterations needed to find a local minimum may be explained by two factors:

A better convergence rate is normally observed with this criterion, compared with the standard one;

The initial value of the criterion (6) is usually much smaller than the initial value of (4), for the same initial value of the nonlinear parameters. This is because at each iteration, including the first, the value of the linear parameters is the optimal, conditioned by the value taken by the nonlinear parameters. As the criterion is very sensitive to the value of the linear parameters, a large reduction is obtained, in comparison with the situation where random numbers are used as starting values for these weights.

2.1Training algorithms

Different training schemes can be employed for the minimization of criteria (4) or (6). Here, only gradient-based completed supervised training algorithms will be considered.

The most used method to minimize (4) is the Error-Back-Propagation (BP) (Werbos, 1974) algorithm, which is a steepest descent algorithm. Each iteration of the BP algorithm is:

$$\boldsymbol{w}[k] = \boldsymbol{w}[k-1] - \eta\boldsymbol{g}[k] \tag{7}$$

where $\boldsymbol{g}$ stands for the gradient vector of (4):

$$\boldsymbol{g}[k] = \frac{\partial}{\partial \boldsymbol{w}^{T}[k]}\Omega(\boldsymbol{w}[k]) \tag{8}$$

Notice that if (6) is used, (7) should be reformulated in terms of $\boldsymbol{v}[k]$, and (8) in terms of ψ and $\boldsymbol{v}[k]$.

The BP algorithm is a first-order method as it only uses derivatives of the first order. If no line-search is used, then it has no guarantee of convergence and the convergence rate obtained is usually very slow. If a second-order method is to be employed, the best algorithm (Gill et. al., 1981) to use is the Levenberg-Marquardt (LM) algorithm (Marquardt, 1963).

Denoting as $\boldsymbol{J}$ the Jacobian matrix:

$$\boldsymbol{J}[k] = \frac{\partial}{\partial \boldsymbol{w}^T[k]} \boldsymbol{o}^{(q)}(\boldsymbol{w}[k]) \tag{9}$$

the LM update, $\Delta[k] = \Omega[k] - \Omega[k-1]$, is given as the solution of:

$$(\boldsymbol{J}^T[k]\boldsymbol{J}[k] + \lambda \boldsymbol{I})\boldsymbol{s}[k] = -\boldsymbol{g}[k] = -\boldsymbol{J}^T[k]\boldsymbol{e}[k] \tag{10}$$

In (10), λ is a regularization parameter, which controls both the search direction and the magnitude of the update. The good results presented by the LM method (compared with other second-order methods such as the quasi-Newton and conjugate gradient methods) are due to its explicit exploitation of the underlying characteristics of the optimization problem, a sum-of-square of errors.

Please notice that, if (6) is used, (9) and (10) should be updated as described before. In order to perform the training, the derivatives of (6) must be obtained. For this purpose the derivative of $\boldsymbol{A}$ w.r.t. $\boldsymbol{v}$ must be introduced. This is a three-dimensional quantity and will be denoted as:

$$(\boldsymbol{A})_{\boldsymbol{v}} = \frac{\partial \boldsymbol{A}}{\partial \boldsymbol{v}} \tag{11}$$

Notice that, for each nonlinear parameter $\boldsymbol{v}_i$, $(\boldsymbol{A})_{\boldsymbol{v}_i}$ is a matrix with only one non-zero column. Considering first the use of the error back-propagation algorithm with this new criterion, the gradient of ψ must be obtained.

It can be proved (see Ruano, 1992) that the gradient vector of ψ can be given as:

$$\begin{bmatrix} 0 \\ \boldsymbol{g}_\psi \end{bmatrix} = \boldsymbol{g}_\Omega\Big|_{\boldsymbol{u}=\hat{\boldsymbol{u}}} = -\begin{bmatrix} (\boldsymbol{o}^{(q-1)})^T \\ 1 \\ \boldsymbol{J}_\psi^T \end{bmatrix} \boldsymbol{e}_\Psi \tag{12}$$

where $\boldsymbol{g}_\Omega\big|_{\boldsymbol{u}=\hat{\boldsymbol{u}}}$, $\boldsymbol{J}_\psi$ and $\boldsymbol{e}_\psi$ denote respectively the gradient, the partition of the Jacobian matrix associated with the weights $\boldsymbol{v}$ and the error vector of Ω obtained when the values of linear weights are their conditional optimal values, i.e.:

$$\boldsymbol{J}_\psi = (\boldsymbol{A})_{\boldsymbol{v}} \boldsymbol{A}^+ \boldsymbol{t} \tag{13}$$

$$\boldsymbol{e}_\psi = \boldsymbol{P}_{\boldsymbol{A}_\perp} \boldsymbol{t} \tag{14}$$

The simplest way to compute $\boldsymbol{g}_\psi$ or $\boldsymbol{J}_\psi$ is to propagate the input patterns through the nonlinear layer(s), compute the optimal values of $\hat{\boldsymbol{u}}$ using (5), and then compute $\boldsymbol{g}_\psi$ or $\boldsymbol{J}_\psi$ using standard algorithms. Notice also that, due to the special structure of $(\boldsymbol{A})_{\boldsymbol{v}_i}$, we only need to compute $\frac{\partial \boldsymbol{O}^{(q-1)}_{\cdot,i}}{\partial \boldsymbol{v}_i}$, where the subscript $\cdot, i$ denotes the i^{th} column of the matrix and $\boldsymbol{v}_i$ denotes the partition of the nonlinear parameters vector related with the i^{th} column of the matrix.

The computation of $\boldsymbol{g}_\psi$ has the additional burden of determining $\hat{\boldsymbol{u}}$. However, since the dimensionality of the problem has been reduced, there is also some gain at each iteration that must be taken into account. Several algorithms can be used to compute these optimal values, but QR or SVD factorization are advisable, because of possible ill-conditioning of $\boldsymbol{A}$.

Although there is an additional cost in complexity per iteration to pay using this approach, a large reduction in the number of iterations needed for convergence is obtained if (6) is used as the training criterion.

If the Levenberg-Marquardt method is employed with this new formulation, then the Jacobian of (6) must be obtained. Three different Jacobian matrices have been proposed for this problem: the first was introduced by (Golub and Pereyra, 1973), who were also the first to introduce the reformulated criterion. To reduce the computational complexity of this approach, in (Kaufman, 1975) a simpler Jacobian matrix is proposed. Ruano *et al.* (Ruano *et al.*, 1991) suggested to employ the Jacobian matrix $\boldsymbol{J}_\psi$, introduced in (12), which further reduces the computational complexity of each training iteration. Notice that the computation of this Jacobian matrix follows the same lines as the computation of $\boldsymbol{g}_\psi$, discussed earlier. In terms of computational costs, the computation of $\boldsymbol{J}_\psi$ does not involve more costs than the computation of $\boldsymbol{J}$. The use of this Jacobian matrix in the LM algorithm with the

reformulated criterion makes each iteration actually less complex than if the standard criterion is used.

To employ these techniques for the different models considered, we must explicitly compute the corresponding derivatives. Only the Jacobian computation will be indicated here, as it is known that the gradient vector can be obtained as:

$$\boldsymbol{g} = -\boldsymbol{J}^T\boldsymbol{e} \tag{15}$$

Multilayer Perceptrons. In this type of networks, the computation of the gradient is sufficiently described in the literature as the BP algorithm (see, for instance, Rumelhart *et al.*, 1986), and for this reason, will be omitted. The Jacobian can be obtained by backpropagating 1, and not the error, and by not performing the accumulation through all patterns in each weight.

Radial Basis Functions. The output of an RBF, with Gaussian activation function for the hidden layers, can be expressed as:

$$\boldsymbol{o}^{(3)}[j] = \boldsymbol{O}^{(2)}\boldsymbol{u} = \left[e^{-\frac{\|\boldsymbol{c}_i - \boldsymbol{x}[j]\|_2^2}{2\sigma_i^2}} \right] \boldsymbol{u} \tag{16}$$

where i runs through all neurons and j through all patterns. An output bias is sometimes considered.

To compute $\frac{\partial \boldsymbol{O}_{j,i}^{(2)}}{\partial \boldsymbol{v}_i}$, as the nonlinear parameters are the centre vector $(\boldsymbol{c}_i)$ and the spread (σ_i) for all neurons, the following equations are used:

$$\frac{\partial \boldsymbol{O}_{j,i}^{(2)}}{\partial \boldsymbol{c}_i} = \boldsymbol{O}_{j,i}^{(2)} \frac{(\boldsymbol{O}_{j,.}^{(1)} - \boldsymbol{c}_i)}{\sigma_i^2} \tag{17}$$

$$\frac{\partial \boldsymbol{O}_{j,i}^{(2)}}{\partial \sigma_i} = \boldsymbol{O}_{j,i}^{(2)} \frac{\left\|\boldsymbol{O}_{j,.}^{(1)} - \boldsymbol{c}_i\right\|_2^2}{\sigma_i^3} \tag{18}$$

Wavelet Networks. Wavelet decomposition is a typical example of the use of local basis functions. Loosely speaking, the "mother basis function" (usually referred as *mother wavelet*) is dilated and translated to form a wavelet basis. Considering, for instance, a unidimensional Mortlet mother wavelet, the network output can be given as:

$$\boldsymbol{o}^{(3)} = \boldsymbol{O}^{(2)}\boldsymbol{u} = \left[\cos(2\pi \boldsymbol{Z}_{j,i}) e^{-\frac{\boldsymbol{Z}_{j,i}^2}{2}} \right] \boldsymbol{u}, \tag{19}$$

where

$$\boldsymbol{Z}_{j,i} = (\boldsymbol{o}_j^{(1)} - \boldsymbol{t}_i)\boldsymbol{d}_i, \tag{20}$$

$\boldsymbol{t}_i$ and $\boldsymbol{d}_i$ being the translation and dilation parameters, respectively. The derivatives are:

$$\frac{\partial \boldsymbol{O}_{j,i}^{(2)}}{\partial \boldsymbol{t}_i} = -\boldsymbol{d}_i \frac{\partial \boldsymbol{O}_{j,i}^{(2)}}{\partial \boldsymbol{Z}_{j,i}} \tag{21}$$

$$\frac{\partial \boldsymbol{O}_{j,i}^{(2)}}{\partial \boldsymbol{t}_i} = -\boldsymbol{d}_i \frac{\partial \boldsymbol{O}_{j,i}^{(2)}}{\partial \boldsymbol{Z}_{j,i}}, \tag{22}$$

where

$$\frac{\partial \boldsymbol{O}_{j,i}^{(2)}}{\partial \boldsymbol{Z}_{j,i}} = -\boldsymbol{Z}_{j,i}\boldsymbol{O}_{j,i}^{(2)} - 2\pi \sin(2\pi \boldsymbol{Z}_{j,i}) e^{-\frac{\boldsymbol{Z}_{j,i}^2}{2}} \tag{23}$$

B-Splines Neural Networks. B-spline neural networks belong to the class of networks termed lattice-based *associative memories networks* (AMN). They are composed of three layers: a *normalized input space layer*, a *basis functions layer* and a *linear weight layer*.

As the complexity of these networks rises exponentially with the number of inputs, it is beneficial to employ, instead of one single multivariate B-spline model covering all the *n*-dimensional input space, a linear combination of sub-models, each one with lower input dimension. The nonlinear parameters, in this network, are the *knots* employed to form a grid in the input space.

Due to lack of space, a detailed description of this network and the computation of its derivatives is omitted. It can be found in (Ruano *et. al*. 2001).

Mamdani Fuzzy Models. It can be proved (see Ruano *et. al*. 2001) that, provided the following assumptions are verified:

The input value x is a crisp numeric quantity;

The linguistic terms of the rule antecedents and consequents are modelled by B-splines of order 2, i.e., triangular membership functions;

For implementing logic connectives such as the conjunction and implication, the t-norm used is the product;

The membership function of the output fuzzy set is computed pointwisely by taking the sum over all the rule's output fuzzy sets;

·The linguistic-to-numeric conversion (the so-called "defuzzification" process) is obtained by the centre of gravity method,

a Mamdani fuzzy model is equivalent to a B-spline network with one multivariate sub-model, where splines of order 2 are used for every dimension. The network basis functions are equivalent to the linguistic terms of the rule antecedents, and the network output linear parameters are related with the linguistic terms of the rule consequents and their confidence values by a linear relationship. Therefore, the training algorithms described are also applicable to Mamdani fuzzy models that satisfy those assumptions.

Takagi-Kang-Sugeno Fuzzy Models. The Takagi-Kang-Sugeno (or simply Sugeno) model is a fuzzy model where the consequents of the rules are real-valued functions; polynomials being the most common, and widely used. For the SISO case, a typical Sugeno model has a set of rules, such as the following:

$$R^{(i)}::=\text{if } (x \text{ is } X^{(i)}) \text{ then } y^{(i)} = w_0^{(i)} + w_1^{(i)} x \qquad (24)$$

$i = 1 \ldots r$, r being the number of rules.

Given a crisp input datum x, the output of this model is:

$$y = \frac{\sum_{i=1}^{r} \mu_{X^{(i)}}(x) y^{(i)}}{\sum_{i=1}^{r} \mu_{X^{(i)}}(x)} \qquad (25)$$

When the linguistic terms of the rule-antecedents $X^{(i)}$ are modelled by B-splines, for all x in the input space, $\sum_{i=1}^{r} \mu_{X^{(i)}}(x) = 1$. Therefore, (25) can be written as:

$$\boldsymbol{o}^{(3)} = \boldsymbol{O}^{(2)} \boldsymbol{u}, \qquad (26)$$

where

$$\boldsymbol{O}^{(2)} = \left[\mu_{X^{(1)}}(x) \ \ldots \ \mu_{X^{(r)}}(x) \ \mu_{X^{(1)}}(x) x \ \ldots \ \mu_{X^{(r)}}(x) x \right] \qquad (27)$$

$$\boldsymbol{u} = \left[w_0^{(1)} \ \ldots \ w_0^{(r)} \ w_1^{(1)} \ \ldots \ w_1^{(r)} \right]^T \qquad (28)$$

This can be easily generalized for the multi-input case.

In terms of derivatives, and denoting by $\left.\frac{\partial \boldsymbol{O}^{(2)}}{\partial \lambda}\right|_{BS}$ the derivatives of the basis functions of a B-spline/ Mamdani sub-model, the derivatives of a TSK model, can be given as:

$$\left.\frac{\partial \boldsymbol{O}^{(2)}}{\partial \lambda}\right|_{TSK} = \left[\left.\frac{\partial \boldsymbol{O}^{(2)}}{\partial \lambda}\right|_{BS} \left.\frac{\partial \boldsymbol{O}^{(2)}}{\partial \lambda}\right|_{BS} \times \boldsymbol{x} \right], \qquad (29)$$

where the symbol $\times$ implies that every column of $\left.\frac{\partial \boldsymbol{O}^{(2)}}{\partial \lambda}\right|_{BS}$ is multiplied, element-by-element, by the vector $\boldsymbol{x}$.

3. RESULTS

Two examples will be employed to demonstrate the application of the reformulated criterion. The first example, an academical problem, illustrates an inverse kinematic transformation between Cartesian coordinates and one of the angles of a two-links manipulator.

Considering first an MLP with 2 hidden layers, with 4 neurons in each layer, and a RBF with 10 hidden neurons, the following table illustrates the results obtained in terms of MSE, for the first 100 iterations of the BP and LM algorithms, minimizing both (4) and (6). The value in brackets in the BP cells denotes the learning rate used.

Table 1: MSE after 100 iterations

	BP (4)	BP (6)	LM (4)	LM (6)
MLP	(0.01) 14 10^{-3}	(0.5) 1 10^{-3}	6.6 10^{-9}	2.7 10^{-9}
RBF	(0.001) 3 10^{-3}	(0.1) 2 10^{-6}	4 10^{-6}	3.5 10^{-9}

Comparing the 2 criteria, it can be concluded that the reformulated criterion produces better results than the standard one, independently of the training algorithm used. If the 2 algorithms are compared, the LM algorithm definitely performs better.

Considering the same example, the following table illustrates the results obtained with a B-spline/ Mamdani model and a TSK model. Both used 2 interior knots, employing triangular splines. Here a termination criterion, related with the desired accuracy in the results was used. For this reason, the following table presents the results of the training algorithms in terms of the number of iterations needed to find a local minimum with the desired accuracy. A maximum value of 100 implies that training was terminated at

that iteration, without convergence. The minimum MSEs for B-spline and TSK were 14 10^{-3} and 9 10^{-4}, respectively. Analysing this table, the same conclusions as above can be taken.

Table 2: Number of iterations

	BP (4)	BP (6)	LM (4)	LM (6)
B-Splines	(.005) 100	(0.01) 21	12	9
TSK	(.005) 100	(0.1) 100	20	13

To illustrate the application of these methods for wavelet networks, an example taken from (Matos *et al*., 2001) is used. Here, wavelet networks are used to approximate transcranial Doppler ultrasound blood flow signals, which will be, in a later stage, classified as normal, or embolic (gaseous or solid) signals.

Employing a wavelet network, with 10 Mortlet neurons, Table 3 illustrates the results obtained, for one particular signal. Here, a termination criterion was also employed, and both the final MSE obtained and the final iteration are reported. As before, a limit of 100 training iterations was considered.

Table 3: Number of iterations and final MSE

	Numb. Iter.	Final MSE
BP (4) - 5 10^{-3}	90	8.9
BP (6)- 5 10^{-4}	100	5.7
LM (4)	100	6.9
LM (6)	25	1.9

4. CONCLUSIONS

In this paper it was made explicit that, in training neural networks and neuro-fuzzy systems, the aim is really to find the basis functions in the last hidden layer. As a consequence of this view, the training criterion should be changed so that it is only dependent on the parameters related with the hidden layers.

This approach can be used with the whole class of gradient-based training algorithms, presenting, usually, better convergence properties. It has been demonstrated with multilayer perceptrons, radial basis functions, wavelet networks, B-splines, Mamdani and TSK fuzzy models, although it is applicable to any kind of structure whose output is a linear combination of nonlinearly parameterized basis functions.

Although, in the authors' experience, the Levenberg-Marquardt algorithm minimizing the reformulated criterion, introduced here, performs always better than any training algorithm specifically introduced for the model considered, it should be stressed that it only guarantees (a fast) convergence to a local minimum. Other types of algorithms, namely evolutionary algorithms, should be used to derive "good" initial nonlinear model parameters, in order to increase the chances of determining the global minimum. These algorithms can also be applied to determine a "good" structure (number of neurons, number of sub-models, etc.) for the neural or neuro-fuzzy model at hand. These topics are the subject of the author's current research.

ACNOWLEDGEMENTS

The authors acknowledge the support of Praxis.

REFERENCES

Gill, P., W. Murray and M. Wright (1981). *Practical optimization*. Academic Press

Golub, G. and V. Pereyra (1973). The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM JNA*, **10**, 2, 413-432

Kaufman, L. (1975). A variable projection method for solving separable nonlinear least squares problems. *BIT*, **15**, 49-57

Marquardt, D. (1963). An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math.*, **11**, 431-441

Matos, S., M.G. Ruano, A.E. Ruano and D.H. Evans (2001). Neural networks calssification of cerebral embolic signals, accepted to *23rd Annual Int. Conf. of the IEEE Engineering in Medicine and Biology Society*, Istanbul, Turkey, 25-28 October

Ruano, A. E., D. Jones, D. and Fleming P. J. (1991). A new formulation of the learning problem for a neural network controller. *30th IEEE Conference on Decision and Control*, Brighton, England, **1**, 865-866

Ruano, A.E. (1992). *Applications of Neural Networks to Control Systems*. PhD. Thesis, UCNW, UK.

Ruano, A. E., C. Cabrita, C., J. V. Oliveira and L.T. Koczy (2001). Completely supervised training algrothms for B-spline and neuro-fuzzy systems, accepted to *IFAC Conference on New Technologies for Computer Control (NTCC'2001)*, Hong-Kong, 19-22 November

Rumelhart, D., McClelland, J. and the PDP Research Group (1986). *Parallel distributed processing* Vol. 1, MIT Press

Sjoberg, J., Q. Zhang, L. Ljung, A. Benveniste, B-Delyon, P. Glorennec, H. Hjalmarsson and A. Juditsky (1995). Nonlinear Black-box Modelling in Systems Identification: a Unified Overview. *Automatica*, **31**, 12, 1691-1724

Werbos, P. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. Doctoral Dissertation, Appl. Math, Harvard University, U.S.A.