TEMPORAL ASSOCIATIVE MEMORY AND FUNCTION APPROXIMATION WITH THE SELF-ORGANIZING MAP

Guilherme de A. Barreto and Aluizio F. R. Araújo
Departmento de Engenharia Elétrica, Universidade de São Paulo (USP)
Av. Trabalhador Sancarlense, 400, 13566-590, São Carlos-SP, Brazil
Phone: +55 16 273 9357, Fax: +55 16 273 9372
E-mail: gbarreto,aluizioa@sel.eesc.sc.usp.br
Web: http://www.sel.eesc.sc.usp.br/ lasi/

Abstract. We propose an unsupervised neural modelling technique, called *Vector-Quantized Temporal Associative Memory* (VQTAM), which enables Kohonen's self-organizing map (SOM) to approximate nonlinear dynamical mappings globally. A theoretical analysis of the VQTAM scheme demonstrates that the approximation error decreases as the SOM training proceeds. The SOM is compared with standard MLP and RBF networks in the forward and inverse identification of a hydraulic actuator. The simulation results produced by the SOM are as accurate as those produced by the MLP network, and better than those produced by the RBF network; both the MLP and the RBF being supervised algorithms. The SOM is also less sensitive to weight initialization than MLP networks. The paper is concluded with a brief discussion on the main properties of the VQTAM approach.

1. INTRODUCTION

Dynamic system identification (or modelling) is the discipline interested in building mathematical models of nonlinear systems, starting from experimental time series data, measurements, or observations [7]. Typically, a certain linear or nonlinear model structure which contains unknown parameters (i.e. one puts forward a certain parameterization) is chosen by the user. The resulting model of a system is often very important for analysis, simulation, prediction, monitoring, diagnosis, and control system design. More recently, artificial neural network models have been successfully applied to the identification and control of a variety of nonlinear dynamical systems [5, 11, 4, 8], such as chemical, economical, biological or tecnological processes (or plants). Such achievements are due to a number of results showing that supervised feedforward architectures, such as the multilayer perceptron (MLP) or the

0-7803-7616-1/02/$17.00 (c) 2002 IEEE

radial basis function (RBF) networks, can approximate arbitrarily well any continuous input-output mapping (see [5] for a review).

In this paper, we propose a system modelling technique which uses unsupervised neural networks for function approximation, instead of the usual supervised ones (MLP and RBF) [11]. This technique, called *Vector-Quantized Temporal Associative Memory* (VQTAM), shows that the Self-Organizing Map (SOM) [6] can be successfully used to approximate nonlinear input-output mappings globally. Computer simulations illustrate this approximation ability of the SOM using the VQTAM approach and compare the obtained results with those produced by MLP and RBF networks and linear models.

The remainder of the paper is organized as follows. In Section 2, we briefly introduce the function approximation problem. Section 3 introduces the VQTAM tecnique and its main properties. It is also shown how the VQTAM can be used together with SOM in the forward and inverse identification problems. Section 4 brings a theoretical analysis of the VQTAM approach. Section 5 compares the SOM with other linear and nonlinear methods in the forward and inverse identification of a hydraulic actuator. The paper is concluded in Section 6.

2. NONLINEAR FUNCTION APPROXIMATION

In this paper, we assume that the systems of interest are governed by the following nonlinear discrete-time difference equation [8]:

$$\mathbf{y}(t+1) = \mathbf{f}[\mathbf{y}(t), \ldots, \mathbf{y}(t-n_y+1); \mathbf{u}(t), \ldots, \mathbf{u}(t-n_u+1)]. \qquad (1)$$

Thus, the system output $\mathbf{y} \in \Re^q$ at time $t+1$ depends, in the sense defined by the nonlinear map $\mathbf{f}(\cdot)$, on the past n_y output values and on the past n_u values of the input $\mathbf{u} \in \Re^p$. In many identification and control problems, it is also desirable to approximate the inverse dynamics of a nonlinear plant[1]:

$$\mathbf{u}(t) = \mathbf{f}^{-1}[\mathbf{y}(t+1), \mathbf{y}(t), \ldots, \mathbf{y}(t-n_y+1); \mathbf{u}(t-1), \ldots, \mathbf{u}(t-n_u+1)] \qquad (2)$$

As pointed out in the introductory section, MLP and RBF networks have been the most commonly used *model structures* with the goal of providing an approximation for the nonlinear function $\mathbf{f}(\cdot)$ or for its inverse(s) $\mathbf{f}^{-1}(\cdot)$, using only the available input-output data, $\{\mathbf{u}(t), \mathbf{y}(t)\}$, $t = 1, \ldots, N$. This occurs in part because the identification problem can be easily set up in terms of a supervised training scheme. Our goal, however, is to devise a strategy that enables the SOM to approximate input-output mappings.

Simply put, the SOM is an unsupervised neural algorithm designed to represent neighborhood (spatial) relationships among vectors of an unlabelled

[1] When used for direct-inverse control, the term $\mathbf{y}(t+1)$ in (2) is replaced by a reference signal, $\mathbf{r}(t+1)$, which is usually assumed to be available at time t [5].

data set [6]. The neurons in the SOM are put together in an output layer, $\mathcal{A}$, in one-, two- or even three-dimensional arrays. Each neuron $i \in \mathcal{A}$ has a weight vector $\mathbf{w}_i \in \Re^n$ with the same dimension as the input vector $\mathbf{x} \in \Re^n$. The SOM algorithm was originally designed to learn *static* input-output mappings [13], usually described by $\mathbf{y}(t) = \mathbf{g}(\mathbf{u}(t))$, in which the current output $\mathbf{y}(t)$ depends solely on the current input $\mathbf{u}(t)$, i.e., no memory for past inputs and outputs is available. For the SOM to be able to learn dynamic mappings, it must have some type of *short-term memory* (STM) mechanism [1, 2, 3]. That is, the SOM should be capable of temporarily storing past information about the system input and output vectors. By using STM mechanisms, such as delay lines, the SOM algorithm is used to approximate the nonlinear function $\mathbf{f}(\cdot)$ or its inverse $\mathbf{f}^{-1}(\cdot)$, as we demonstrate next.

3. TEMPORAL ASSOCIATIVE MEMORY USING THE SOM

In order to approximate nonlinear dynamic mappings, the input vector to the SOM, $\mathbf{x}(t)$, is augmented so that it has from now onwards two parts. The first part, denoted as $\mathbf{x}^{in}(t)$, carries data about the input of the dynamic mapping being learned. The second part, denoted $\mathbf{x}^{out}(t)$, contains data concerning the desired output of this mapping. The weight vector of neuron i has its dimension increased accordingly. These changes are written as:

$$\mathbf{x}(t) = \begin{pmatrix} \mathbf{x}^{in}(t) \\ \mathbf{x}^{out}(t) \end{pmatrix} \quad \text{and} \quad \mathbf{w}_i(t) = \begin{pmatrix} \mathbf{w}_i^{in}(t) \\ \mathbf{w}_i^{out}(t) \end{pmatrix} \tag{3}$$

Depending on the variables chosen to build the vectors $\mathbf{x}^{in}(t)$ and $\mathbf{x}^{out}(t)$ one can use the SOM algorithm to learn the forward or the inverse dynamics of a nonlinear plant. For example, if one wishes to approximate the forward dynamics in (1) the following definitions apply:

$$\mathbf{x}^{in}(t) = [\mathbf{y}(t), \ldots, \mathbf{y}(t - n_y + 1); \mathbf{u}(t), \ldots, \mathbf{u}(t - n_u + 1)] \tag{4}$$

$$\mathbf{x}^{out}(t) = \mathbf{y}(t+1) \tag{5}$$

If the interest is in inverse identification, then we define:

$$\mathbf{x}^{in}(t) = [\mathbf{y}(t+1), \mathbf{y}(t), \ldots, \mathbf{y}(t - n_y + 1); \mathbf{u}(t-1), \ldots, \mathbf{u}(t - n_u + 1)] \tag{6}$$

$$\mathbf{x}^{out}(t) = \mathbf{u}(t) \tag{7}$$

During training, the winning neurons are found using only the portion corresponding to $\mathbf{x}^{in}(t)$:

$$i^*(t) = \arg\min_{i \in \mathcal{A}} \{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\} \tag{8}$$

In updating the weights, both $\mathbf{x}^{in}(t)$ and $\mathbf{x}^{out}(t)$ are used:

$$\Delta \mathbf{w}_i^{in}(t) = \alpha(t) h(i^*, i; t)[\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)] \tag{9}$$

$$\Delta \mathbf{w}_i^{out}(t) = \alpha(t) h(i^*, i; t)[\mathbf{x}^{out}(t) - \mathbf{w}_i^{out}(t)] \tag{10}$$

where $\alpha(t)$ is the learning rate and $h(i^*, i; t)$ is a Gaussian neighborhood function given by $h(i^*, i; t) = \exp\left(-\|\mathbf{r}_i(t) - \mathbf{r}_{i^*}(t)\|^2/\sigma^2(t)\right)$, where $\mathbf{r}_i(t)$ and $\mathbf{r}_{i^*}(t)$ are, respectively, the locations of neurons i and i^* in the output array. The variables $\alpha(t)$ and $\sigma(t)$ decay exponentially with time: $\alpha(t) = \alpha_0\,(\alpha_T/\alpha_0)^{(t/T)}$ and $\sigma(t) = \sigma_0\,(\sigma_T/\sigma_0)^{(t/T)}$, where α_0 and σ_0 denote their initial values, while α_T and σ_T are the final ones, after T training iterations.

Note that the learning rules in (9) and (10) follow the usual formulation of the SOM: the first rule acts on the input and the second one acts on the output of the mapping being learned. The underlying idea is that, as training proceeds, the SOM algorithm learns to associate, in a topology-preserving way, the outputs $\mathbf{x}^{out}(t)$ of the mapping with the corresponding inputs $\mathbf{x}^{in}(t)$. Thus, this technique will be referred to as *Vector-Quantized Temporal Associative Memory* (VQTAM). It is also worth emphasizing the difference between this unsupervised strategy and that one used for training supervised networks. In MLP and RBF networks, the vector $\mathbf{x}^{in}(t)$ is presented to the network input, while the $\mathbf{x}^{out}(t)$ is used at the network output to compute an error signal that guides learning. In other words, supervised networks use *error-based learning*, that is, they calculate an explicit error between the output they generate and the desired output (teaching signal), and learning is designed to reduce the error. The VQTAM scheme instead allows unsupervised neural networks, such as the SOM, to learn to *associate* inputs and outputs, and the learning mechanism essentially calculates different forms of associations or correlations.

A trained SOM network can then be used to obtain estimates for the output of the learned mapping from the output portion of the weight vector, $\mathbf{w}_i^{out}(t)$, as follows:

$$\hat{\mathbf{y}}(t+1) \equiv \mathbf{w}_{i^*}^{out}(t) \;\text{ (forward case)} \quad \text{or} \quad \hat{\mathbf{u}}(t) \equiv \mathbf{w}_{i^*}^{out}(t) \;\text{ (inverse case)} \tag{11}$$

where in both cases the winning neuron, $i^*(t)$, is found as defined in (8). The estimation process continues for M steps until an entirely new series is built from the estimated values.

4. MATHEMATICAL ANALYSIS OF THE VQTAM

The goal of this section is to present a mathematical analysis of the convergence process of the SOM algorithm using the VQTAM approach. More specifically, we want to know if the additional learning rule in (10) really allows the SOM to learn an input-output mapping asymptotically. This analysis, based on the stochastic approximation by Robbins and Monro [10], aims at showing that the estimation (or approximation) error decreases as the SOM training proceeds, converging eventually to a stable state. For clarity's sake, we assume that the function to be approximated is given as in (1) and the only source of information available is the time series of measured input-output data, $\{\mathbf{u}(t), \mathbf{y}(t)\}$, $t = 1, \ldots, N$. As shown in Section 3, the SOM can be used to provide an approximation, $\hat{F}(\cdot)$, of the unknown nonlinear

function $\mathbf{f}(\cdot)$ using an associative memory approach. In this case, the SOM provides an estimate $\hat{\mathbf{y}}(t+1)$, given an unforeseen vector $\mathbf{x}^{in}(t)$, as follows:

$$\hat{\mathbf{y}}(t+1) \equiv \mathbf{w}_{i^*}^{out}(t) = \hat{F}[\mathbf{x}^{in}(t); i^*(t)] \tag{12}$$

where the explicit representation of the winning neuron $i^*(t)$ in (12) emphasizes the associative nature of the VQTAM approach, in which $i^*(t)$ is the element responsible for ***linking*** the input and output portions of the learned mapping. Then, we define an absolute value for the estimation error:

$$\varepsilon(t) = \|\mathbf{y}(t+1) - \hat{\mathbf{y}}(t+1)\| \equiv \|\mathbf{x}^{out}(t) - \mathbf{w}_{i^*}^{out}(t)\| \tag{13}$$

Through the definition of this error measure we can now prove that the proposed unsupervised function approximation scheme is equivalent to an ***implicit*** error-based learning procedure.

One of the main features of the SOM algorithm is the cooperation between the winning neuron $i^*(t)$ and its neighbors through the neighborhood function $h(i^*, i; t)$. Without loss of generality, we assume that the neighborhood function is time-invariant. To take into account this cooperation during the function approximation process, we define the following quantity:

$$J(t) = \sum_{i=1}^{L} h(i^*, i; t)\varepsilon^2(t) = \sum_{i=1}^{L} h(i^*, i; t)\|\mathbf{x}^{out}(t) - \mathbf{w}_i^{out}(t)\|^2 \tag{14}$$

where L is the number of neurons in the SOM. Thus, the network convergence can be evaluated through the mean expected squared estimation error as defined by the following functional:

$$\tilde{J} = E\left\{\sum_{i\in L} h(i^*, i)\varepsilon^2\right\} = \int_{\mathbf{X}} \sum_{i\in L} h(i^*, i)\|\mathbf{x}^{out} - \mathbf{w}_i^{out}\|^2 p(\mathbf{X})d\mathbf{X} \tag{15}$$

where it is assumed that the expectation value is taken over an infinite sequence of stochastic samples $\mathbf{X}=\{\mathbf{x}^{out}(t)\}$, $t = 1, 2, \ldots$; $p(\mathbf{X})$ means the joint probability density function of the whole input sequence $\mathbf{X}$ over which the estimator is formed, and $d\mathbf{X}$ is a "volume differential" in the space in which $\mathbf{X}$ is formed. The time index t is omitted for clarity in (15) because it is defined implicitly in the temporal order of the sequence $\mathbf{X}$.

Since the probability density function $p(\mathbf{X})$ is unknown and the sequence $\mathbf{X}$ is finite in reality, we shall resort to Robbins-Monro stochastic approximation technique for the minimization of $\tilde{J}$, i.e., to find an *optimal* value, $\tilde{\mathbf{w}}_i^{out}$, for the parameter $\mathbf{w}_i^{out}$. The basic idea is to try to decrease the function $J(t)$ at each new step t by descending in the direction of its negative gradient with respect to the current parameter vector $\mathbf{w}_i^{out}(t)$. The recursive formula for the parameter vector $\mathbf{w}_i^{out}$ reads

$$\mathbf{w}_i^{out}(t+1) = \mathbf{w}_i^{out}(t) - \frac{1}{2}\alpha(t)\frac{\partial J(t)}{\partial \mathbf{w}_i^{out}(t)} \tag{16}$$

where the scalar $\alpha(t)$ defines the step size and satisfies $\sum_{t=0}^{\infty} \alpha(t) = \infty$ and $\sum_{t=0}^{\infty} \alpha^2(t) < \infty$. Considering (14), the partial derivative in (16) is given by:

$$\frac{\partial J(t)}{\partial \mathbf{w}_i^{out}(t)} = -2h(i^*, i; t)[\mathbf{x}^{out}(t) - \mathbf{w}_i^{out}(t)] \tag{17}$$

The recursive equation in (16) is then written as:

$$\mathbf{w}_i^{out}(t+1) = \mathbf{w}_i^{out}(t) + \alpha(t)h(i^*, i; t)[\mathbf{x}^{out}(t) - \mathbf{w}_i^{out}(t)] \tag{18}$$

Equation (18) is exactly the learning rule in (10). Hence, starting from arbitrary initial values, $\mathbf{w}_i^{out}(0)$, the sequence $\{\mathbf{w}_i^{out}(t)\}$ will converge to the neighborhood of the optimal vector $\bar{\mathbf{w}}_i^{out}$.

5. SIMULATIONS

Figure 1 shows measured values of the valve size (input variable), $u \in \Re$, and the oil pressure (output variable), $y \in \Re$, of a hydraulic actuator. As can be seen in the oil pressure time series, there are very oscillative behaviors caused by mechanical resonances. These data have been used in benchmarking studies on nonlinear system identification [11]. The SOM, MLP and RBF networks are used as nonlinear identification models to approximate the forward and the inverse nonlinear dynamics of the hydraulic actuator. In the forward modelling task, these three neural nets are also compared with the usual linear *Autoregressive model with Exogenous Inputs* (ARX): $\hat{y}(t+1) = \sum_{i=0}^{n_y-1} a_i y(t-i) + \sum_{j=0}^{n_u-1} b_j u(t-j)$, where a_i and b_j are the coefficients of the model and $\hat{y}(t+1)$ is the estimated value for the plant output at time step $t+1$. The coefficients are computed by the Least-Squares Method [7]. The approximation accuracy is evaluated through the root mean square error: $RMSE = \sqrt{\frac{1}{M}\sum_{t=0}^{M-1}(o(t) - \mathbf{w}_{i^*}^{out}(t))^2}$ where $o(t) = y(t+1)$ or $o(t) = u(t)$ depending on the mapping being learned and M is the length of the estimated series. The data are presented to the four models without any preprocessing stage. A total number of $N = 1024$ samples are available for both the input and output variables. The first 512 samples are used to train the three networks and to compute the coefficients of the linear ARX model, while the remaining 512 samples are used to validate the four models. A training epoch is defined as one presentation of the training samples. For all the simulations, it is assumed $n_y = 3$ and $n_u = 2$, as suggested in [11].

An one-dimensional SOM is simulated in this paper. This network has six input units, since $\dim(\mathbf{x}^{in}) + \dim(\mathbf{x}^{out}) = 5 + 1 = 6$, and 500 output neurons. The weights are randomly initialized between 0 and 1, and adjusted for 600 epochs[2]. The training parameters are the following: $\alpha_0 = 1.0$, $\alpha_T = 10^{-5}$, $\sigma_0 = 250$, $\sigma_T = 10^{-3}$ and $T = 600 \times 512 = 3 \times 10^5$. The best result

[2]Other ranges for random weight initialization, e.g., between -1 and 1 have been tested but they had no significant influence on the final configuration of the weights.

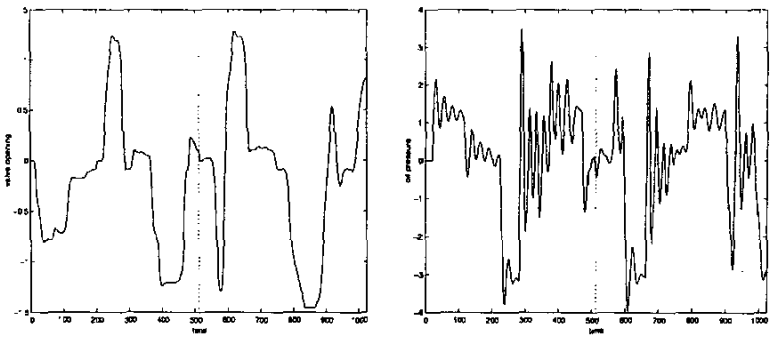

Figure 1: Measured values of valve position (left) and oil pressure (right).

generated by the SOM model during validation ($RMSE = 0.2051$), together with an evaluation of the influence of the number of training epochs in the approximation performance, are shown in Figure 2. One can note that the error decays very fast initially, stabilizing around RMSE=0.20 after just 100 epochs. The results provided by the linear ARX model were not very good ($RMSE = 1.0133$).

The MLP network has five input units since $\dim(\mathbf{x}^{in}) = 5$, one hidden layer with ten neurons and one output neuron. The neurons in the hidden layer have hyperbolic tangent transfer functions, while the output neuron has a linear transfer function. The MLP network is trained with backpropagation algorithm with momentum. The values for the learning rate and the momentum factor are set to 0.2 and 0.9, respectively. The training is stopped if $RMSE \leq 0.001$ or a maximum number of 600 training epochs is reached. The RBF network also has five input units, an intermediate layer with neurons with Gaussian basis function, and one output neuron. Following the

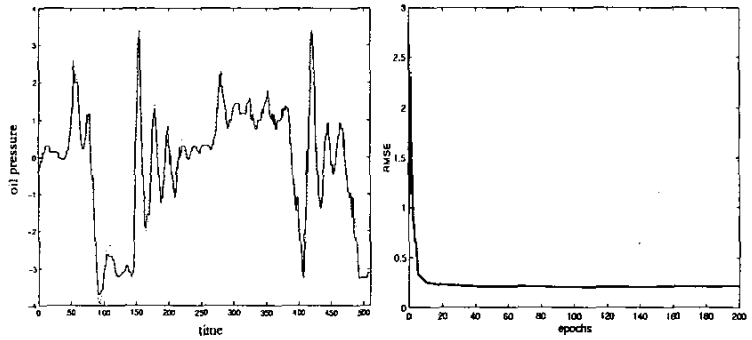

Figure 2: Simulation of the SOM (left) model on validation data and evolution of the forward approximation error with the number of training epochs (right). Solid line: simulated signal. Dotted line: true oil pressure.

Table 1: RMSE values for the MLP, SOM, WTA and RBF networks in the forward and inverse identification of the hydraulic actuator.

	Forward Identification				Inverse Identification			
RMSE	*Min*	*Max*	*Mean*	$\sqrt{var}$	*Min*	*Max*	*Mean*	$\sqrt{var}$
MLP	0.1162	0.2493	0.1554	0.0457	0.0566	0.4789	0.1446	0.1259
SOM	0.2051	0.2665	0.2259	0.0215	0.1189	0.1255	0.1207	0.0021
WTA	0.3199	0.6157	0.4073	0.0993	0.2664	0.2996	0.2733	0.0097
RBF	0.2067	0.4103	0.2994	0.0774	0.1398	0.2841	0.2032	0.0551

RBF design in [12], which is specific for regression problems, the number of neurons in the intermediate layer is the same as the number of training samples, and hence there is a Gaussian kernel centered at every training vector. The intermediate-to-output weights are just the target values, so the output is simply a weighted average of the target values of training cases close to the given input case. The only weights that need to be learned are the radii of the Gaussian kernels. In this paper, this parameter was the same for all RBF units, being deterministically varied to evaluate its effect in the accuracy of the approximation. In order to evaluate the influence of the neighborhood function $h(i^*, i; t)$ on the final approximation results, we also simulated a pure winner-take-all (WTA) network. This is equivalent to training the SOM without updating the weights of the neurons in the neighborhood of the winner, i.e., using $h(i^*, i; t) = 1$ if $i = i^*(t)$, and $h(i^*, i; t) = 0$, otherwise. The results for the four networks in the forward and inverse identification of the hydraulic actuator are shown in Table 1. The RMSE values shown for the SOM, WTA and MLP networks were averaged over ten training runs. For the RBF the radius was varied from 0.1 (minimum RMSE) to 1.0 (maximum RMSE) in increments of 0.1.

One can note that the MLP network provides the best results in general. The SOM algorithm, in its turn, produces better results than a RBF network with approximately the same number of neurons. The minimum RMSE for the RBF network occurs for radius=0.1, increasing in an approximate linear fashion to a maximum for radius=1.0. An interesting result is that the SOM is less sensitive to weight initialization than the MLP network, as can be seen in the fifth column of Table 1. This sensitivity is measured through the standard deviation of the RMSE values generated for the 10 training runs. Furthermore, the SOM can adapt on-line to new incoming data without retraining the entire network as in the standard MLP case. Another difficulty found in designing the MLP network is the occurrence of overtraining, an issue related to the choice of the number of hidden neurons, which results in poor performance (high RMSE) during model validation. Since the SOM network is in essence a type of vector quantization algorithm: the more the SOM network is trained, the more precise is the approximation (in a statistical sense) of the probability distribution of the training data [6]. However, after some training time, learning in the SOM stabilizes around RMSE=0.20

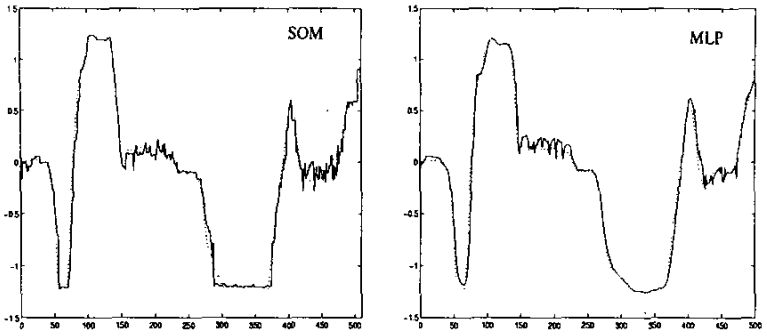

Figure 3: Results obtained by the SOM (left) and by the MLP (right) for the inverse identification of the hydraulic actuator.

(Figure 2) and no substantial reduction in RMSE is noted. The results for the WTA network illustrate that the neighborhood cooperation is in fact decisive for a good performance of the VQTAM approach. The best results produced by the SOM and MLP networks in the inverse identification of the hydraulic actuator are shown in Figure 3. For these specific cases, the estimation errors were $RMSE(\text{SOM}) = 0.1189$ and $RMSE(\text{MLP}) = 0.0566$. It is worth noting that in some portions of the estimated time series the errors had high values. This occurs because the inverse mapping may not be ***unique***, an issue that makes it inherently harder to approximate than the forward mapping.

6. DISCUSSION AND CONCLUSION

Function approximation through neural networks is a research field dominated completely by supervised learning paradigms. Only recently the use of unsupervised neural networks for dynamic system modelling and control has become more common (see [9]). The main differences between the method in [9] and the VQTAM is that the latter learns an input-output mapping globally, in the sense that only one model structure is used; while the former uses several (linear) model structures, each one responsible for modelling a localized region of the system's state space. In general, local modelling with the SOM produces slightly better results than global modelling. However, local modelling demands higher computational efforts than global modelling, an issue that may become important in real-time control. Another issue of interest is that the MLP and RBF networks define a continuous input-output mapping, while the SOM builds a discrete input-output mapping with an inherent quantization error. We are currently working on unsupervised strategies to building a continuous input-output mapping through a continuous version of the SOM [13], thus reducing the quantization error.

The simulations shown in this paper illustrate the potential of the VQTAM technique. Additional tests should be performed, such as residual anal-

ysis of the estimation error, noise and fault tolerance, to demonstrate effectively the viability of using the SOM algorithm in identification and control of nonlinear dynamic systems. Currently, research is being conducted with the aim of designing a predictive nonlinear controller using the SOM and the VQTAM approach. A quantitative comparison with local modelling schemes are also being performed.

Acknowledgements

The authors thank FAPESP (a Brazilian research agency) for its financial support under the grant Nr. 98/12699-7.

REFERENCES

[1] A. F. R. Araújo and G. A. Barreto, "Context in temporal sequence processing: A self-organizing approach and its application to robotics," **IEEE Transactions on Neural Networks**, vol. 13, no. 1, pp. 45–57, 2002.

[2] G. A. Barreto and A. F. R. Araújo, "Time in self-organizing maps: An overview of models," **International Journal of Computer Research**, vol. 10, no. 2, pp. 139–179, 2001.

[3] B. deVries and J. Principe, "The Gamma model – A new neural model for temporal processing," **Neural Networks**, vol. 5, no. 4, pp. 565–576, 1992.

[4] S. Haykin and J. Principe, "Making sense of a complex world: Using neural networks to dynamically model chaotic events such as sea clutter," **IEEE Signal Processing Magazine**, vol. 15, no. 3, pp. 66–81, 1998.

[5] K. J. Hunt, D. Sbarbaro, R. Zbikowski and P. J. Gawthrop, "Neural networks for control systems – A survey," **Automatica**, vol. 28, no. 6, pp. 1083–1112, 1992.

[6] T. Kohonen, **Self-Organizing Maps**, Berlin, Heidelberg: Springer-Verlag, 2nd edn., 1997.

[7] L. Ljung and T. Glad, **Modeling of Dynamic Systems**, Englewood Cliffs, NJ: Prentice-Hall, 1994.

[8] M. Norgaard, O. Ravn, N. K. Poulsen and L. K. Hansen, **Neural Networks for Modelling and Control of Dynamic Systems**, Springer-Verlag, 2000.

[9] J. Principe, L. Wang and M. Motter, "Local dynamic modeling with self-organizing maps and applications to nonlinear system identification and control," **Proceedings of the IEEE**, vol. 86, no. 11, pp. 2240–2258, 1998.

[10] H. Robbins and S. Monro, "A stochastic approximation method," **Annals of Mathematical Statistics**, vol. 22, pp. 400–407, 1951.

[11] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Deylon, P.-Y. Glorennec, H. Hjalmarsson and A. Juditsky, "Nonlinear black-box modeling in system identification: A unified overview," **Automatica**, vol. 31, no. 12, pp. 1691–1724, 1995.

[12] D. F. Specht, "A Generalized Regression Neural Network," **IEEE Transactions on Neural Networks**, vol. 2, no. 5, pp. 568–576, 1991.

[13] J. Walter and H. Ritter, "Rapid learning with parametrized self-organizing maps," **Neurocomputing**, vol. 12, pp. 131–153, 1996.