Novel Algorithms for Nonlinear Channel Equalization Using Neural Vector Quantization

Luís G. M. Souza, Guilherme A. Barreto and João C. M. Mota

***Abstract*— In this paper we use the Self-Organizing Map (SOM), a well-known neural vector quantization algorithm, to design nonlinear adaptive filters through the *Vector-Quantized Temporal Associative Memory* (VQTAM) method. In VQTAM, the centroids (codebook vectors) of input clusters found by the SOM are associated with codebook vectors of output clusters, so that the SOM can learn dynamic input-output mappings in a very simple and effective way. In addition, we also propose two VQTAM-based Radial Basis Function (RBF) adaptive filters. Firstly, a global RBF model is built using all the input codebook vectors as centers of M gaussian basis functions, while the hidden-to-output layer weights are given by the output prototypes. Then, a local RBF model is built in a similar fashion, but using only $K \ll M$ neurons. We evaluate the proposed VQTAM-based adaptive filters in a nonlinear channel equalization task. Performance comparisons with the standard linear FIR/LMS and the nonlinear *Multilayer Perceptron* (MLP) equalizers are also carried out.**

***Index Terms*— Self-Organizing Maps, Vector Quantization, Radial Basis Functions, Channel Equalization.**

***Resumo*— Neste trabalho, usa-se a Rede Auto-Organizável de Kohonen (SOM, sigla em inglês), um conhecido algoritmo neural de quantização vetorial, para projetar filtros adaptativos não-lineares por meio do método de Memória Associativa Temporal por Quantização Vetorial (VQTAM, sigla em inglês). Neste método, os vetores-código (protótipos) dos dados de entrada encontrados pela rede SOM são associados com os vetores-códigos dos dados de saída, permitindo que a SOM aprenda mapeamentos dinâmicos entrada-saída de modo simples e efetivo. Além disso, dois filtros adaptativos baseados na arquitetura da rede de Funções de Base Radial (RBF) e no método VQTAM são propostos. Primeiramente, um modelo RBF Global é construído usando todos os vetores-código de entrada como centros de M funções de base gaussiana, enquanto os pesos da camada de saída são obtidos a partir dos protótipos da saída. Em seguida, um modelo RBF local é construído de forma similar, porém usando somente $K \ll M$ neurônios. Os filtros adaptativos propostos são avaliados na equalização de um canal não-linear. Comparações de desempenho com equalizador linear (FIR/LMS) e um não-linear (rede *Perceptron Multicamadas*) são também realizadas.**

***Palavras-Chave*— Mapas Auto-Organizáveis, Quantização Vetorial, Funções de Base Radial, Equalização de Canal.**

I. Introduction

Modern telecommunications impose severe quality requirements for radio link or cellular telephony systems. Crucial for attaining such requirements are the design of adaptive channel equalization algorithms capable to deal with several adverse effects, such as noise, intersymbol interference (ISI), co-channel and adjacent channel interference, nonlinear distortions, fading, time-varying characteristics, among others [8], [18].

Supervised neural networks, such as MLP and RBF, motivated by their universal approximation property, have been successfully used as nonlinear tools for channel equalization [11]. It has been demonstrated that the performance of MLP- or RBF-based adaptive filters usually outperform traditional linear techniques in many common signal processing applications [12], [20], [25].

The *Self-Organizing Map* (SOM) is an important unsupervised neural architecture which, in contrast to the supervised ones, has been less applied to adaptive filtering. For being a kind of clustering algorithm, its main field of application is vector quantization of signals [9], [10], and hence, it is not used as a stand-alone function approximator, but rather in conjunction with standard linear [19], [23] or nonlinear [4] models.

Recently, the *Vector-Quantized Temporal Associative Memory* (VQTAM) [1] method was proposed to enable the SOM to learn input-output mappings, such as those commonly encountered in time series prediction, robotics and control system applications. In these tasks the performance of the SOM was comparable to or better than those of supervised neural architectures.

In this paper we further explore the function approximation ability of the VQTAM in difficult adaptive filtering tasks, such as nonlinear channel equalization. Through simulations we demonstrate the superior performance of the VQTAM-based adaptive filters over the standard linear and nonlinear adaptive structures, such as the FIR/LMS, the MLP and the *Least-Squares SOM-based* (LESSOM) local regression model proposed in [17].

The remainder of the paper is organized as follows. Section II summarizes the VQTAM technique and illustrates how it can be applied to channel equalization. In Section III we show how to build global and local RBF models through VQTAM. In Section IV several computer simulations evaluate the performance of the proposed VQTAM-based adaptive filters in nonlinear channel equalization. The paper is concluded in Section V.

II. The VQTAM Methodology

The VQTAM method is based on the SOM algorithm, which can be defined as an unsupervised neural algorithm designed to build a representation of neighborhood (spatial) relationships among vectors of an unlabeled data set [13]–[15]. Neurons in the SOM are put together in an output layer, $\mathcal{A}$, arranged

Luís G. M. Souza, Guilherme A. Barreto and João C. M. Mota, Department of Teleinformatics Engineering, Federal University of Ceará (UFC), Fortaleza, Ceará, Brazil, E-mails: {luisgustavo, guilherme, mota}@deti.ufc.br.

in one-, two- or even three-dimensional array (grid). The arrangement of the neurons in this well-defined geometric grid are necessary in order to define a physical neighborhood for each neuron in the grid.

Each neuron $i \in \mathcal{A}$ has a weight vector $\mathbf{w}_i \in \Re^n$, also called *codebook vector*, with the same dimension of the input vector $\mathbf{x} \in \Re^n$. The network weights are trained according to a competitive-cooperative scheme in which the weight vectors of a winning neuron and its neighbors in the output array are updated after the presentation of each input vector. This training procedure eventually leads the neurons to reproduce proximity relationships of the input data vectors in their weight vectors. That is, data vectors that are close in the data space are mapped to neighboring neurons in the SOM array. This property is called *topology preservation* and is one of the main strength of the SOM over conventional vector quantization (clustering) algorithms.

The VQTAM method itself is a generalization to the temporal domain of a SOM-based associative memory technique that has been used by many authors to learn static (memoryless) input-output mappings, specially within the domain of robotics (see [2] and references therein). In both cases, the input vector to be presented to the SOM at time step t, $\mathbf{x}(t)$, is composed of two parts. The first part, denoted $\mathbf{x}^{in}(t) \in \mathbb{R}^p$, carries data about the input of the dynamic mapping to be learned. The second part, denoted $\mathbf{x}^{out}(t) \in \mathbb{R}^q$, contains data concerning the desired output of this mapping. The weight vector of neuron i, $\mathbf{w}_i(t)$, has its dimension increased accordingly. These changes are formulated as follows:

$$\mathbf{x}(t) = \begin{pmatrix} \mathbf{x}^{in}(t) \\ \mathbf{x}^{out}(t) \end{pmatrix} \quad \text{and} \quad \mathbf{w}_i(t) = \begin{pmatrix} \mathbf{w}_i^{in}(t) \\ \mathbf{w}_i^{out}(t) \end{pmatrix} \qquad (1)$$

where $\mathbf{w}_i^{in}(t) \in \mathbb{R}^p$ and $\mathbf{w}_i^{out}(t) \in \mathbb{R}^q$ are, respectively, the portions of the weight (codebook) vector which store information about the inputs and the outputs of the mapping being learned.

Depending on the variables chosen to build the vectors $\mathbf{x}^{in}(t)$ and $\mathbf{x}^{out}(t)$ one can use the SOM to learn forward or inverse mappings. It is worth emphasizing that these vectors not necessarily have the same dimensionality. Indeed, we have $p > q$ in general.

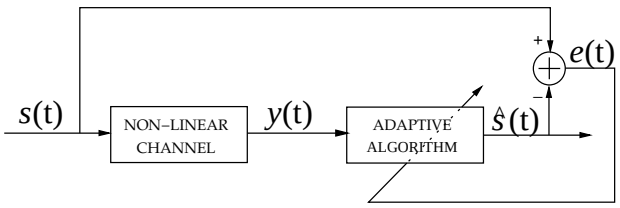
In this paper, we are interested in the equalization of nonlinear channels, a complex adaptive filtering task corresponding to the learning of the inverse model of the channel (see Fig. 1a). In this case, we have $p > 1$ and $q = 1$, and the following definitions apply:

$$\mathbf{x}^{in}(t) = [y(t) \;\; y(t-1) \;\; \cdots \;\; y(t-p+1)]^T \qquad (2)$$
$$\mathbf{x}^{out}(t) = s(t-\tau) \qquad (3)$$

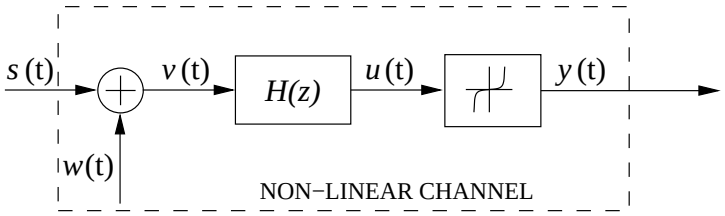
where $s(t)$ is the symbol transmitted at the time step t, $y(t)$ is the corresponding channel output, $\tau \geq 0$ is the symbol delay, p is the order of the equalizer, and the superscript T denotes the transpose vector. Without loss of generality, in this paper we assume $\tau = 0$.

During learning, the winning neuron at time step t is determined based only on $\mathbf{x}^{in}(t)$:

$$i^*(t) = \arg\min_{i \in \mathcal{A}} \{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\} \qquad (4)$$



(a)



(b)

Fig. 1. (a) Channel equalization by an adaptive filter and (b) the block diagram of the noisy nonlinear channel used in the simulations.

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

$$\mathbf{w}_i^{in}(t+1) = \mathbf{w}_i^{in}(t) + \alpha h(i^*, i; t)[\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)] \qquad (5)$$
$$\mathbf{w}_i^{out}(t+1) = \mathbf{w}_i^{out}(t) + \alpha h(i^*, i; t)[\mathbf{x}^{out}(t) - \mathbf{w}_i^{out}(t)] \qquad (6)$$

where $0 < \alpha < 1$ is the learning rate and $h(i^*, i; t)$ is a time-varying Gaussian neighborhood function defined as follows:

$$h(i^*, i; t) = \exp\left(-\frac{\|\mathbf{r}_i(t) - \mathbf{r}_{i^*}(t)\|^2}{2\gamma^2(t)}\right) \qquad (7)$$

where $\mathbf{r}_i(t)$ and $\mathbf{r}_{i^*}(t)$ are the coordinates of the neurons i and i^* in the SOM array, respectively. The parameter $\gamma(t)$ defines the width of the neighborhood of the current winning neuron. For the sake of convergence of the SOM, the parameter $\gamma(t)$ should decrease exponentially in time according to the following equation:

$$\gamma(t) = \gamma_0 \left(\frac{\gamma_N}{\gamma_0}\right)^{t/N} \qquad (8)$$

where γ_0 and γ_N are the initial and final values of the neighborhood, and N is the length of the training sequence.

In words, the learning rule in (5) performs the topology-preserving vector quantization of the input space and the rule in (6) acts similarly on the output space of the mapping being learned. As training proceeds, the SOM learns to associate the input codebook vectors $\mathbf{w}_i^{in}$ with the corresponding output codebook vectors $\mathbf{w}_i^{out}$.

The SOM-based associative memory procedure implemented by the VQTAM can then be used for function approximation purposes. More specifically, once the SOM has been trained, its output $\mathbf{z}(t)$ for a new input vector is estimated from the learned codebook vectors, $\mathbf{w}_{i^*}^{out}(t)$, as follows:

$$\mathbf{z}(t) \equiv \mathbf{w}_{i^*}^{out}(t) \qquad (9)$$

where $\mathbf{w}_{i^*}^{out} = [w_{1,i^*}^{out} \;\; w_{2,i^*}^{out} \;\; \cdots \;\; w_{q,i^*}^{out}]^T$ is the weight vector of the current winning neuron $i^*(t)$, found as in (4). For the channel equalization task we are interested in, we have set $q = 1$. Thus, the output of the VQTAM-based equalizer is a scalar version of (9), given by:

$$z(t) = \hat{s}(t) = w_{1,i^*}^{out}(t) \tag{10}$$

where $\hat{s}(t)$ is the estimated transmitted symbol at time t.

It is worth noting that this kind of associative memory is different from the usual supervised approach. 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 explicitly an error signal that guides learning. The VQTAM method instead allows competitive neural networks, such as the SOM, to correlate the inputs and outputs of the mapping without computing an error signal explicitly[1].

III. Building Efficient RBF Models from VQTAM

The VQTAM method itself can be used for function approximation purposes. However, since it is essentially a vector quantization method, it may require a large number of neurons to achieve an accurate generalization. To improve its performance, we introduce two RBF models obtained from a SOM network trained under the VQTAM scheme.

A. A Global RBF Model

Assuming that the SOM has N_h neurons, a general Radial Basis Function network with N_h gaussian basis functions and q output neurons can be built over the learned input and output codebook vectors, $\mathbf{w}_i^{in}$ and $\mathbf{w}_i^{out}$, as follows:

$$\mathbf{z}(t) = \frac{\sum_{i=1}^{N_h} \mathbf{w}_i^{out} G_i(\mathbf{x}^{in}(t))}{\sum_{i=1}^{N_h} G_i(\mathbf{x}^{in}(t))} \tag{11}$$

where $\mathbf{z}(t) = [z_1(t) \;\; z_2(t) \;\; \cdots \;\; z_q(t)]^T$ is the output vector, $\mathbf{w}_i^{out} = [w_{1,i}^{out} \;\; w_{2,i}^{out} \;\; \cdots \;\; w_{q,i}^{out}]^T$ is the weight vector connecting the ith basis function to the q output units, and $G_i(\mathbf{x}^{in}(t))$ is the response of this basis function to the current input vector $\mathbf{x}^{in}(t)$, i.e.

$$G_i(\mathbf{x}^{in}(t)) = \exp\left(-\frac{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}\|^2}{2\sigma^2}\right) \tag{12}$$

where $\mathbf{w}_i^{in}$ plays the role of the center of the ith basis function and σ defines its radius (or $spread$).

Note that in (11), all the N_h codebook vectors are used to estimate the corresponding output. In this sense, we referred to the RBF model just described as the *Global* RBF (GRBF) model, despite the localized nature of each gaussian basis.

Since we are interested in the equalization of a single communication channel, we set $q = 1$. In this case, the output vector of the GRBF network in (11), which is used to estimate (recover) the current transmitted symbol, reduces to a scalar output, $z(t)$, defined as:

$$z(t) = \hat{s}(t) = \frac{\sum_{i=1}^{N_h} w_{1,i}^{out} G_i(\mathbf{x}^{in}(t))}{\sum_{i=1}^{N_h} G_i(\mathbf{x}^{in}(t))} \tag{13}$$

[1]In reality, an error signal is computed $implicitly$ in (6).

In the usual two-phase RBF training, the centers of the basis functions are firstly determined by clustering the $\mathbf{x}^{in}$ vectors (e.g. using K-means algorithm) and then the hidden-to-output layer weights are then computed through the LMS rule (or the pseudo-inverse method). In the GRBF model just described one single learning phase is necessary, in which SOM-based clustering is performed simultaneously on the input-output pairs $\{\mathbf{x}^{in}(t), \mathbf{x}^{out}(t)\}$.

It is worthwhile to contrast the GRBF with two well-known RBF design strategies, namely the *Generalized Regression Neural Network* (GRNN) [21] and the *Modified Probabilistic Neural Network* (MPNN) [24]. In the GRNN, there is a basis function centered at every training input data vector $\mathbf{x}^{in}$, and the hidden-to-output weights are just the target values $\mathbf{x}^{out}$. The MPNN and the GRNN share the same theoretical background and basic network structure. The difference is that MPNN uses the K-means clustering method for the computation of its centers.

For both the GRNN/MPNN models, the output is simply a weighted average of the target values of training vectors close to the given input vector. For the GRBF instead, the output is the weighted average of the output codebook vectors $\mathbf{w}_i^{out}$ associated with the input codebook vectors $\mathbf{w}_i^{out}$ close to the given input vector $\mathbf{x}^{in}$, as stated in (11) and (13).

B. A Local RBF Model

Recently, many authors have studied the problem of approximating nonlinear input-output mappings through the so-called *local models* [3], [5], [16], [17]. According to this approach just a small portion of the modeled input-output spaces are used to estimate the output for a new input vector.

In the context of the VQTAM approach, local modeling means that we need only $1 < K < N_h$ prototypes to set up the centers of the basis functions and the hidden-to-output weights of a RBF model. To this purpose, we suggest to use the weight vectors the first K winning neurons, denoted by $\{i_1^*, i_2^*, \ldots, i_K^*\}$, which are determined as follows:

$$\begin{aligned}
i_1^*(t) &= \arg\min_{\forall i} \left\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}\|\right\} \\
i_2^*(t) &= \arg\min_{\forall i \neq i_1^*} \left\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}\|\right\} \\
&\vdots \qquad\qquad \vdots \qquad\qquad \vdots \\
i_K^*(t) &= \arg\min_{\forall i \neq \{i_1^*, \ldots, i_{K-1}^*\}} \left\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}\|\right\}
\end{aligned} \tag{14}$$

The estimated output is now given by:

$$z(t) = \frac{\sum_{k=1}^{K} w_{1,i_k^*}^{out} G_{i_k^*}\left(\mathbf{x}^{in}(t)\right)}{\sum_{i_k^*=1}^{K} G_{i_k^*}\left(\mathbf{x}^{in}(t)\right)} \tag{15}$$

We referred to the local RBF model thus built as the *KRBF* model. It is worth noting that the VQTAM and the GRBF become particular instances of the KRBF if we set $K = 1$ and $K = N_h$, respectively.

A local RBF model was proposed earlier in [6]. First, a GRNN model is built and then only those centers within a certain distance $\varepsilon > 0$ from the current input vector are used to estimate the output. This idea is basically the same as that

used to build the KRBF, but it suffers from the same drawbacks of the GRNN model regarding its high computational cost. Furthermore, depending on the value of ε the number of selected centers may vary considerably (in a random way) at each time step. If ε is too small, it may happen that no centers are selected at all! This never occurs for the KRBF model, since the same number of K centers is selected at each time step.

More recently another local RBF model was proposed in [7] based on a double vector quantization procedure. This method associates a small cluster of input-output pairs $\{\mathbf{x}^{in}(t), \mathbf{x}^{out}(t)\}$ to each neuron, so that N_h local RBF models are built, one for each neuron. In the KRBF only a single local RBF model is built dynamically from the K prototype vectors closest to the current input vector.

IV. Simulations

To evaluate the proposed VQTAM-based adaptive filters in channel equalization tasks, we simulated a nonlinear noisy channel with memory (Figure 1b). First, a linear channel is realized by the following equations:

$$u(t) = \frac{\mathbf{h}^T \mathbf{v}(t)}{\|\mathbf{h}\|}, \qquad t = 1, 2, \ldots, N$$

where $u(t) \in \mathbb{R}$ is the channel output, $\mathbf{v}(t) = [v(t) \quad v(t-1) \quad \cdots \quad v(t-n+1)]^T$ is the tapped-delay vector containing the n most recent noisy symbols, $\mathbf{h} \in \mathbb{R}^n$ is the linear channel impulse response, and N is the length of the symbol sequence.

A given noisy symbol at time t is defined as

$$v(t) = s(t) + w(t) \tag{16}$$

where $s(t) \in \mathbb{R}$ is the noise-free transmitted symbol, and $w(t) \sim \mathcal{N}(0, \sigma_w^2)$ is a white gaussian noise sequence. We assume that data sequence $\{s(t)\}_{t=1}^N$ and the noise sequence $\{w(t)\}_{t=1}^N$ are jointly independent.

The symbol sequence $\{s(t)\}_{t=1}^N$ is realized as a first-order Gauss-Markov process, as follows:

$$s(t) = as(t-1) + b\varepsilon(t) \tag{17}$$

where $\varepsilon(t) \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ is the white gaussian driving noise, and $|a| < 1$ for the sake of stationarity. Therefore, the symbol samples are zero-mean gaussian $s(t) \sim N(0, \sigma_s^2)$, with power σ_s^2 given by

$$\sigma_s^2 = \frac{b^2}{1-a^2}\sigma_\varepsilon^2 \tag{18}$$

For $a = 0$, the source signal becomes a white gaussian noise sequence.

Finally, the output $y(t)$ of the nonlinear channel is obtained by applying a static nonlinearity to the signal $u(t)$:

$$\begin{aligned} y(t) &= 0.2u(t) - 0.2u^2(t) + 0.04u^3(t) + \\ &+ 0.9\tanh[u(t) - 0.1u^2(t) + \\ &\quad +0.5u^3(t) - 0.1u^4(t) + 0.5] \end{aligned} \tag{19}$$

In the following simulations we compare the equalizers implemented via the VQTAM, GRBF, KRBF and MLP neural models. In addition, we also evaluate the performance of the conventional FIR/LMS equalizer and the LESSOM model [17]. The LESSOM model is similar to the KRBF since both models use the codebook vectors of K neurons to build a predictor for the transmitted symbol. However, the LESSOM implements a local linear predictor, while the KRBF implements a nonlinear one.

The following parameters were used to set up the simulations: $a = 0.95$, $b = 0.1$, $\sigma_\varepsilon^2 = 1$, $\sigma_w^2 = 0.03$, $\mathbf{h} = [1 \quad 0.8 \quad 0.5]^T$, $p = 5$ and $N = 6000$. Without loss of generality, we assume that the symbol period is larger than the processing time of the proposed adaptive algorithms. Furthermore, no encoding/decoding technique is used for the transmitted symbol sequence.

The equalizers were trained online using the first half of the sequences $\{s(t), y(t)\}$, while the second half was used to test their generalization (prediction) performances. A total of 500 training/testing runs were performed for a given filter in order to assign statistical confidence to the computation of the *Normalized Mean squared error* (NMSE) to build learning and prediction error curves. For each training/testing run, different white noise sequences $\{w(t), \varepsilon(t)\}$ are generated to built new realizations of the signal sequences $\{s(t), y(t)\}$.

The MLP equalizer has a single layer of N_h hidden neurons, all of them with hyperbolic tangent activation functions, and a linear output neuron. Weights are updated through the backpropagation algorithm with momentum term. The learning rate of all equalizers was set to $\alpha = 10^{-1}$, and held constant during training. For the VQTAM model, the parameters of the neighborhood function were set to $\gamma_0 = 0.5N_h$ and $\gamma_N = 10^{-2}$.

The first simulation evaluates the learning curves of the VQTAM, MLP and FIR/LMS equalizers. The results are shown in Figure 2. For this simulation, the number of neurons was set to $N_h = 10$ and the adaptation step size of the FIR/LMS equalizer was set to 10^{-4}. As expected, the nonlinear equalizers performed much better (i.e. faster convergence to lower levels of the error) than the linear one. The VQTAM equalizer converged as fast as the MLP equalizer, but the former has produced slightly higher NMSE values as training proceeds. Despite of this result, the next simulation illustrates that, depending on the number of neurons, the VQTAM and its variants (GRBF and KRBF) are able to perform better than the MLP equalizer during the prediction phase.

The second simulation attempts to evaluate empirically how the number of hidden neurons influences the prediction performance the MLP, VQTAM and GRBF equalizes. The generalization performance of a given equalizer, for each value of N_h varying from 1 to 20, is evaluated in terms of the resulting *normalized mean-squared error* (NMSE). The results are shown in Figures 3a and 3b.

In both figures the MLP performed better when few hidden neurons are used. However, unlike the VQTAM and GRBF models, its performance does not improve as N_h increases. The minimum NMSE for the MLP was obtained for $N_h = 9$. From this value on, the NMSE tends to increase due to overfitting. For the VQTAM and GRBF models, the NMSE always decreases since we are using more codebook vectors to quantize the input-output spaces, thus reducing the associated

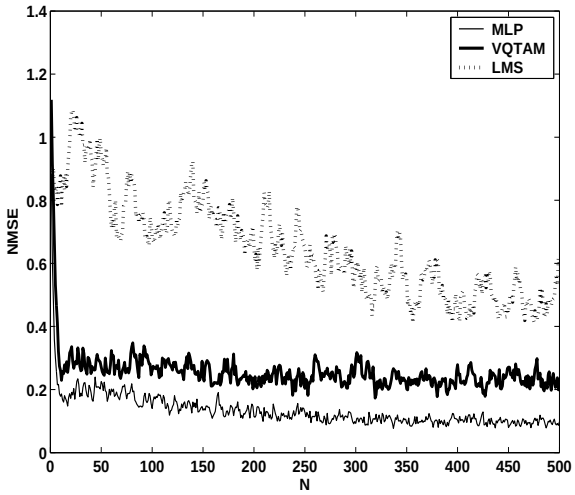


Fig. 2. Learning curves of the MLP and VQTAM equalizers ($N_h = 10$) and the conventional FIR/LMS equalizer.

TABLE I

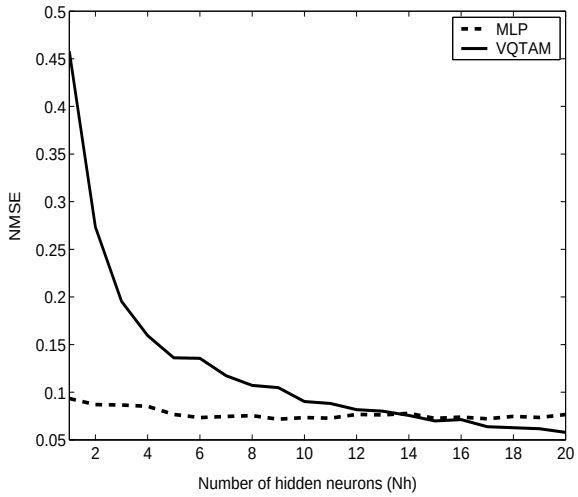
GENERALIZATION PERFORMANCE OF THE SEVERAL EQUALIZERS FOR $N_h = 20$.

Neural Equalizer	**NMSE**			
	mean	*min*	*max*	*variance*
KRBF ($K = 8$)	**0,0487**	0,0190	0,1723	$4{,}67\times10^{-4}$
GRBF	**0,0500**	0,0177	0,2198	$6{,}79\times10^{-4}$
VQTAM	**0,0583**	0,0265	0,1428	$3{,}33\times10^{-4}$
MLP	**0,0785**	0,0185	1,3479	0,0065
LESSOM ($K = 15$)	**0,0991**	0,0251	0,7721	0,0076

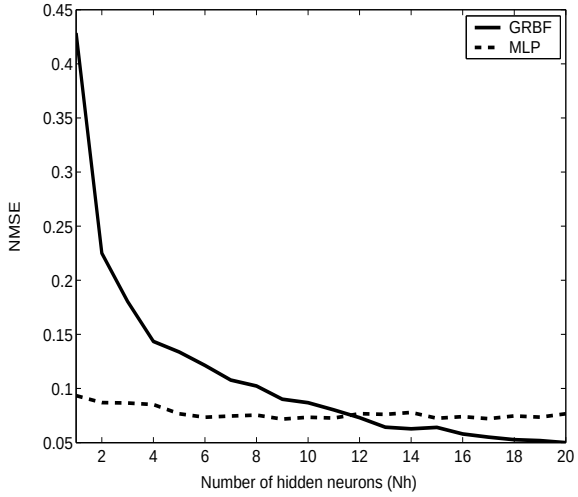
quantization errors. For $N_h \geq 16$, the VQTAM performs better than the MLP. For $N_h \geq 12$, the GRBF performs better than the MLP and VQTAM equalizers, demonstrating that the nonlinear transformation implemented by the gaussian basis functions yields additional prediction power to the GRBF equalizer.

In the third simulation, we evaluate the prediction performance of the local models (KRBF and LESSOM) as a function of the number of prototype vectors used to build the local models. For each input vector, the LESSOM builds the corresponding local model by selecting the prototype vectors of the winning neuron and its $(K-1)$ closest neighboring neurons. These prototypes are then used to set up a linear least-squares regression model. For this simulation, we used $N_h = 20$ and varied K from 8 to 20. The results are shown in Figure 4. One can easily note that the KRBF performed better than the LESSOM for all the range of variation of K. For this simulation, the minimum NMSE for the KRBF was obtained for $K = 8$, while for the LESSOM the minimum was obtained for $K = 15$.

Finally, in Table I we show the best results obtained for the equalizers simulated in this paper, assuming $N_h = 20$. It is worth noting that the KRBF and the GRBF models performed better than the other equalizers and presented the lowest variances in the results.



(a)



(b)

Fig. 3. Prediction error (NMSE) versus the number of hidden neurons (N_h): (a) MLP $\times$ VQTAM, and (b) MLP $\times$ GRBF.

V. CONCLUSION

In this paper we used the Self-Organizing Map (SOM), a well-known neural vector quantization algorithm, to design nonlinear adaptive filters through the *Vector-Quantized Temporal Associative Memory* (VQTAM) method. In addition, we also propose two VQTAM-based Radial Basis Function (RBF) adaptive filters. It was demonstrated through simulations that the proposed models performed better than the linear FIR/LMS equalizer and neural network based equalizers, such as the MLP and the LESSOM model, in nonlinear channel equalization tasks.

We are currently developing learning strategies to determine in an adaptive fashion the number of neurons (N_h) that yields optimal performance for the VQTAM-based equalizers. One of these strategies uses the growing self-organizing map [22] to

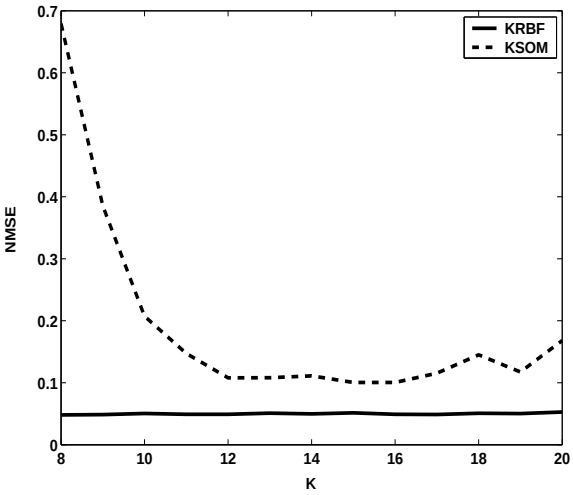


Fig. 4. Prediction error (NMSE) versus the number of local prototypes (K: KRBF $\times$ LESSOM.

add neurons to the network until a given performance criterium has been reached.

ACKNOWLEDGMENT

The authors would like to thank CNPq (grant #305275/2002-0) and FUNCAP (grant #1068/04) for their financial support.

REFERENCES

[1] G. A. Barreto and A. F. R. Araújo, "Identification and control of dynamical systems using the self-organizing map," *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1244–1259, 2004.

[2] G. A. Barreto, A. F. R. Araújo, and H. J. Ritter, "Self-organizing feature maps for modeling and control of robotic manipulators," *Journal of Inteligent and Robotic Systems*, vol. 36, no. 4, pp. 407–450, 2003.

[3] G. A. Barreto, J. C. M. Mota, L. G. M. Souza, and R. A. Frota, "Nonstationary time series prediction using local models based on competitive neural networks," *Lecture Notes in Computer Science*, vol. 3029, pp. 1146–1155, 2004.

[4] S. Bouchired, M. Ibnkahla, D. Roviras, and F. Castanie, "Equalization of satellite mobile communication channels using combined self-organizing maps and RBF networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'98)*, vol. 6, 1998, pp. 3377–3379.

[5] J.-Q. Chen and Y.-G. Xi, "Nonlinear system modeling by competitive learning and adaptive fuzzy inference system," *IEEE Transactions on Systems, Man, and Cybernetics-Part C*, vol. 28, no. 2, pp. 231–238, 1998.

[6] E.-S. Chng, H. H. Yang, and W. Skarbek, "Reduced complexity implementation of the bayesian equaliser using local RBF network for channel equalisation problem," *Electronics Letters*, vol. 32, no. 1, pp. 17–19, 1996.

[7] S. Dablemont, G. Simon, A. Lendasse, A. Ruttiens, F. Blayo, and M. Verleysen, "Time series forecasting with SOM and local non-linear models - Application to the DAX30 index prediction," in *Proceedings of the 4th Workshop on Self-Organizing Maps, (WSOM)'03*, 2003, pp. 340–345.

[8] S. Haykin, *Adaptive Filter Theory*, 4th ed. Prentice-Hall, 2001.

[9] A. Hirose and T. Nagashima, "Predictive self-organizing map for vector quantization of migratory signals and its application to mobile communications," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1532–1540, 2003.

[10] T. Hofmann and J. Buhmann, "Competitive learning algorithms for robust vector quantization," *IEEE Transactions on Signal Processing*, vol. 46, no. 6, pp. 1665–1675, 1998.

[11] M. Ibnkahla, "Applications of neural networks to digital communications – a survey," *Signal Processing*, vol. 80, no. 7, pp. 1185–1215, 2000.

[12] D. Jianping, N. Sundararajan, and P. Saratchandran, "Communication channel equalization using complex-valued minimal radial basis function neural networks," *IEEE Transactions on Neural Networks*, vol. 13, no. 3, pp. 687–696, 2002.

[13] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[14] ——, "The self-organizing map," *Neurocomputing*, vol. 21, pp. 1–6, 1998.

[15] T. Kohonen, E. Oja, O. Simula, A. Visa, and J. Kangas, "Engineering applications of the self-organizing map," *Proceedings of the IEEE*, vol. 84, no. 10, pp. 1358–1384, 1996.

[16] J. Lan, J. Cho, D. Erdogmus, J. Principe, M. Motter, and J. Xu, "Local linear PID controllers for nonlinear control," *International Journal of Control and Intelligent Systems*, vol. 33, 2005.

[17] J. C. Principe, L. Wang, and M. A. 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.

[18] J. G. Proakis, *Digital Communications*. New York: McGraw-Hill, 2001.

[19] K. Raivio, J. Henriksson, and O. Simula, "Neural detection of QAM signal with strongly nonlinear receiver," *Neurocomputing*, vol. 21, no. 1–3, pp. 159–171, 1998.

[20] M. Solazzia, A. Uncini, E. D. Di Claudio, and R. Parisi, "Complex discriminative learning bayesian neural equalizer," *Signal Processing*, vol. 81, no. 12, pp. 2493–2502, 2001.

[21] D. F. Specht, "A general regression neural network," *IEEE Transactions on Neural Networks*, vol. 2, no. 6, pp. 568–576, 1991.

[22] T. Villmann and H.-U. Bauer, "Applications of the growing self-organizing map," *Neurocomputing*, vol. 21, pp. 91–100, 1998.

[23] X. Wang, H. Lin, J. Lu, and T. Yahagi, "Detection of nonlinearly distorted M-ary QAM signals using self-organizing map," *IEICE Transactions on Fundamentals of Electronics, Communications, and Computer Sciences*, vol. E84-A, no. 8, pp. 1969–1976, 2001.

[24] A. Zaknich, "Introduction to the modified probabilistic neural network for general signal processing applications," *IEEE Transactions on Signal Processing*, vol. 46, no. 7, pp. 1980–1990, 1998.

[25] A. Zerguine, A. Shafi , and M. Bettayeb, "Multilayer perceptron-based DFE with lattice structure," *IEEE Transactions on Neural Networks*, vol. 12, no. 3, pp. 532–545, 2001.