Using the Self-Organizing Map to Design Efficient RBF Models for Nonlinear Channel Equalization

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

Abstract - *In this paper we show how to build global and local RBF models once the Self-Organizing Map has been trained using the* Vector-Quantized Temporal Associative Memory *(VQTAM) method. Through the VQTAM, prototype vectors (centroids) of input clusters are associated with prototype vectors of output clusters, so that the SOM can learn dynamic input-output mappings in a very simple and effective way. Global RBF models are built using all the input prototypes as centers of M gaussian basis functions, while the hidden-to-output layer weights are given by the output prototypes. Local RBF models are build in a similar fashion, but using only $K \ll M$ neurons. We evaluate the proposed RBF models and other global/local neural models in a complex nonlinear channel equalization task.*

Key words - Self-Organizing Maps, RBF and MLP Models, Channel Equalization.

1 Introduction

Modern high-speed 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 [1]. Supervised neural networks, such as MLP and RBF, motivated by their universal approximation property, have been successfully used as nonlinear tools for channel equalization. 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 [2].

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 [3], and hence, it is not used as a stand-alone function approximator, but rather in conjunction with standard linear or nonlinear models [4]. Recently, the *Vector-Quantized Temporal Associative Memory* (VQTAM) [5] method was proposed to enable the SOM to learn input-output dynamical 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 to design global and local RBF models for

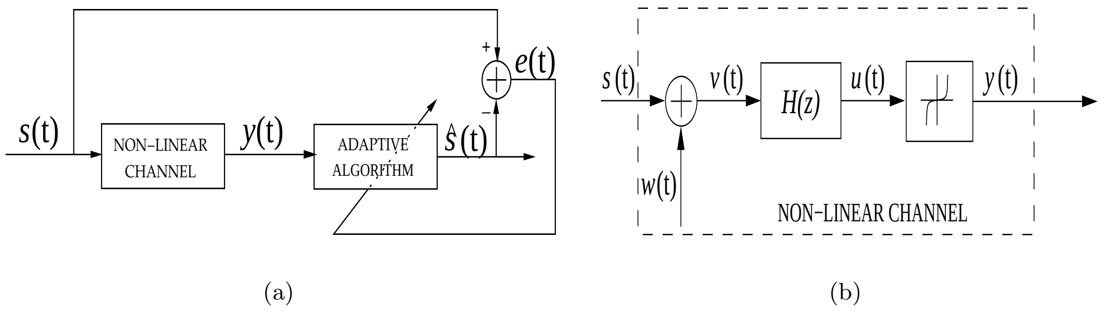


Figure 1: (a) Channel equalization by an adaptive filter and (b) a noisy nonlinear channel model.

adaptive filtering. By simulations we demonstrate the superior performance of the VQTAM-based RBF models, when applied to nonlinear channel equalization, over the MLP and the *Least-Squares SOM-based* (LESSOM) local regression model proposed in [6].

The remainder of the paper is organized as follows. Section 2 summarizes the VQTAM technique. In Section 3 we show how to build global and local RBF models through VQTAM. In Section 4 several computer simulations evaluate the performance of the VQTAM-based RBF networks in nonlinear channel equalization. The paper is concluded in Section 5.

2 The VQTAM Approach

The VQTAM generalizes to the temporal domain 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. In both cases, the input vector to be presented to the SOM, $\mathbf{x}(t)$, 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 $\mathbf{x}^{out}(t) \in \mathbb{R}^q$, $p \geq q$, contains data concerning the desired output of this mapping. The weight vector of neuron i, $\mathbf{w}_i(t) \in \mathbb{R}^{p+q}$, has its dimension increased accordingly. These changes are formulated as follows:

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

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 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. In this paper, we are interested in nonlinear channel equalization, 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 $q = 1$, so that the following definitions apply:

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

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, we assume $\tau = 0$.

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

$$i^*(t) = \arg\min_{i \in \mathcal{A}} \{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)\|\} \tag{3}$$

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

$$\mathbf{w}_i^{in}(t+1) = \mathbf{w}_i^{in}(t) + \alpha h(i^*, i; t)[\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}(t)] \tag{4}$$

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

where $0 < \alpha < 1$ is the learning rate, $h(i^*, i; t) = \exp\left(-\|\mathbf{r}_i(t) - \mathbf{r}_{i^*}(t)\|^2 / 2\gamma^2(t)\right)$ defines a Gaussian-type neighborhood function, for which $\mathbf{r}_i(t)$ and $\mathbf{r}_{i^*}(t)$ are the coordinates of neurons i and i^* in the SOM array, respectively. The neighborhood decreases in time according to $\gamma(t) = \gamma_0 \left(\gamma_N / \gamma_0\right)^{(t/N)}$, where γ_0 and γ_N are their initial and final values, respectively, and N is the length of the training sequence.

In words, the learning rule in (4) performs the topology-preserving vector quantization of the input space and the rule in (5) 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 $\mathbf{w}_i^{out}$.

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:

$$\mathbf{z}(t) \equiv \mathbf{w}_{i^*}^{out}(t) \tag{6}$$

where $\mathbf{w}_{i^*}^{out} = [w_{1,i^*}^{out} \quad w_{2,i^*}^{out} \quad \cdots \quad w_{q,i^*}^{out}]^T$ is the weight vector of the current winning neuron $i^*(t)$, found as in (3). 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 (6), given by:

$$z(t) = \hat{s}(t) = w_{1,i^*}^{out}(t) \tag{7}$$

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

3 Building Efficient RBF Models from VQTAM

The VQTAM method itself can be used for function approximation purposes. However, since it is basically 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.

3.1 A Global RBF Model

For a trained SOM with 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:

$$\mathbf{z}(t) = \mathbf{z}\left(\mathbf{x}^{in}(t), \mathbf{w}_i^{out}, \mathbf{w}_i^{in}\right) = \frac{\sum_{i=1}^{N_h} \mathbf{w}_i^{out} G_i(\mathbf{x}^{in}(t))}{\sum_{i=1}^{N_h} G_i(\mathbf{x}^{in}(t))} \tag{8}$$

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 $G_i(\mathbf{x}^{in}(t)) = \exp\left(-\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}\|^2/2\sigma^2\right)$ is the response of this basis function to the current input vector $\mathbf{x}^{in}(t)$, where $\mathbf{w}_i^{in}$ plays the role of the center of the ith basis function and σ defines its spread.

Note that in (8), 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. Also, since we have set $q = 1$, the output vector of the GRBF network in (8), which is used to recover the current transmitted symbol, reduces to a scalar output, $z(t)$, defined as:

$$z(t) = \hat{s}(t) = \frac{\sum_{i=1}^{N_h} w_{1,i}^{out} G_i(\mathbf{x}^{in}(t))}{\sum_{i=1}^{N_h} G_i(\mathbf{x}^{in}(t))} \tag{9}$$

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 pseudoinverse 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 worth contrasting the GRBF with two well-known RBF design strategies, namely the *Generalized Regression Neural Network* (GRNN) [7] and the *Modified Probabilistic Neural Network* (MPNN) [8]. 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 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, the output is the weighted average of the output prototypes $\mathbf{w}_i^{out}$ associated with the input prototypes $\mathbf{w}_i^{out}$ close to the given input vector $\mathbf{x}^{in}$, as stated by Eq. (8).

3.2 A Local RBF Model

Several authors have studied the problem of approximating nonlinear input-output mappings through *local models* [6, 9]. According to this approach just a small portion of the modelled input/output spaces are used to estimate the output for a new input vector.

In the context of the VQTAM approach, local modelling 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:

$$\begin{aligned}
i_1^*(t) &= \arg\min_{\forall i} \left\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}\|\right\} \equiv \arg\max_{\forall i} \left\{G_i(\mathbf{x}^{in}(t))\right\} \\
i_2^*(t) &= \arg\min_{\forall i \neq i_1^*} \left\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}\|\right\} \equiv \arg\max_{\forall i \neq i_1^*} \left\{G_i(\mathbf{x}^{in}(t))\right\} \\
\vdots \quad & \quad\quad \vdots \quad\quad\quad\quad \vdots \\
i_K^*(t) &= \arg\min_{\forall i \neq \{i_1^*, \ldots, i_{K-1}^*\}} \left\{\|\mathbf{x}^{in}(t) - \mathbf{w}_i^{in}\|\right\} \equiv \arg\max_{\forall i \neq \{i_1^*, \ldots, i_{K-1}^*\}} \left\{G_i(\mathbf{x}^{in}(t))\right\}
\end{aligned} \tag{10}$$

The estimated output is now given by:

$$z(t) = \frac{\sum_{k=1}^{K} w_{1,i_k^*}^{out} G_{i_k^*}(\mathbf{x}^{in}(t))}{\sum_{k=1}^{K} G_{i_k^*}(\mathbf{x}^{in}(t))} \tag{11}$$

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 [10]. 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 [11] 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.

4 Simulations

To evaluate the proposed equalizers, 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.

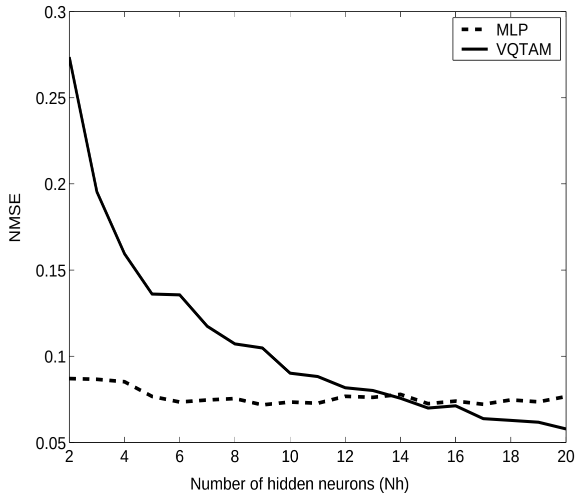
A given noisy symbol at time t is defined as $v(t) = s(t) + w(t)$, where $s(t) \in \mathbb{R}$ is the transmitted (noise-free) symbol, $w(t) \sim \mathcal{N}(0, \sigma_w^2)$ is a white gaussian noise sample. We assume that data sequence $\{s(t)\}_{t=1}^N$ and the noise sequence $\{w(t)\}_{t=1}^N$ are jointly independent. The symbol sequence $\{s(t)\}_{t=1}^N$ is realized as a first-order Gauss-Markov process:

$$s(t) = as(t-1) + b\varepsilon(t) \tag{12}$$

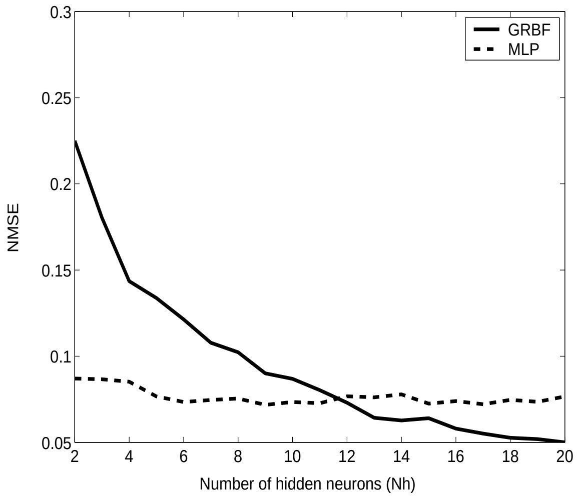
where $\varepsilon(t) \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ is the white gaussian driving noise, and $|a| < 1$ for the sake of stationarity. Thus, the symbol sequence is zero-mean gaussian $s(t) \sim N(0, \sigma_s^2)$, with power $\sigma_s^2 = \frac{b^2}{1-a^2}\sigma_\varepsilon^2$. For $a = 0$, the source signal becomes a white gaussian noise sequence.

Finally, the output of the nonlinear channel, $y(t)$, is obtained as follows:

$$\begin{aligned} y(t) \quad = \quad & 0.2u(t) - 0.2u^2(t) + 0.04u^3(t) + \\ & +0.9 \tanh\left(u(t) - 0.1u^2(t) + 0.5u^3(t) - 0.1u^4(t) + 0.5\right) \end{aligned} \tag{13}$$



(a)



(b)

Figure 2: MSE versus the number of hidden neurons: (a) MLP $\times$ VQTAM, and (b) MLP $\times$ GRBF.

In the simulations we compare VQTAM-, GRBF-, KRBF-, MLP- and LESSOM-based equalizers. The following parameters were used: $a = 0.95$, $b = 0.1$, $\sigma_\varepsilon^2 = 1$, $\sigma_w^2 = 0.03$, $\mathbf{h} = [1 \quad 0.8 \quad 0.5]^T$, $p = 5$ and $N = 6000$. The equalizers were trained online using the first 5000 elements of the sequences $\{s(t), y(t)\}$, while the remaining 1000 elements were used to test their generalization performances. A total of 500 training/testing runs were performed for a given equalizer in order to assign statistical confidence to the computation of the *Normalized Mean squared error* (NMSE) to build prediction (generalization) 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 $\alpha = 10^{-2}$, 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 attempts to evaluate empirically how the number of hidden neurons influences the generalization (prediction) performance the MLP, VQTAM and GRBF models in the equalization task. 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 values of NMSE. The results are shown in Figures 2a and 2b.

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 MSE for the MLP was obtained for $N_h = 9$. From this value on, the MSE tends to increase due to overfitting. For the VQTAM and GRBF models, the MSE tends to decrease since we are using more prototypes to quantize the input/output spaces, thus reducing the associated quantization errors. For $N_h \geq 16$, the VQTAM performs better than

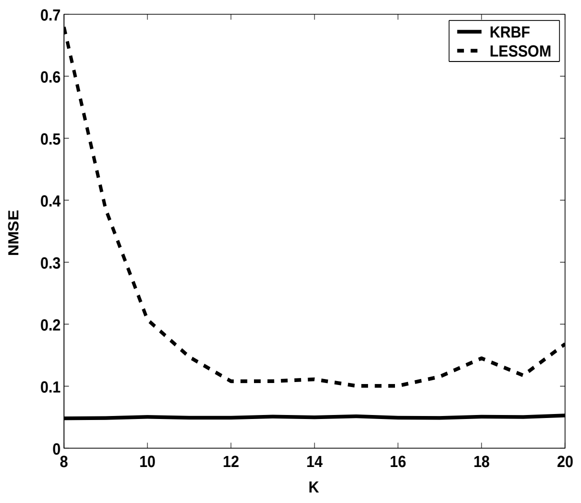


Figure 3: MSE versus the number of local prototypes: KRBF $\times$ LESSOM.

the MLP. For $N_h \geq 12$, the GRBF performs better than the MLP and VQTAM equalizers. Note that the chosen type of channel nonlinearity favours the MLP equalizer, explaining partially why the MLP had a better performance for a small number of hidden neurons.

In the second 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 3. 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 MSE for the KRBF was obtained for $K = 8$, while for the LESSOM the minimum was obtained for $K = 15$.

Finally, in Table 1 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.

5 Conclusion

In this paper we introduced global and local RBF models built over the weight vectors of a Self-Organizing Map trained under the *Vector-Quantized Temporal Associative Memory* method. It was demonstrated that the proposed models performed better than the MLP and the LESSOM model in nonlinear channel equalization tasks.

Acknowledgements: The authors thank CNPq (project #305275/2002-0) and FUNCAP (project #1068/04) for their financial support.

Table 1: Generalization performance of the several equalizers for $N_h = 20$.

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

References

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

[2] M. Ibnkahla. Applications of neural networks to digital communications – a survey. *Signal Processing*, 80(7):1185–1215, 2000.

[3] 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*, 14(6):1532–1540, 2003.

[4] 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*, E84-A(8):1969–1976, 2001.

[5] G. A. Barreto and A. F. R. Araújo. Identification and control of dynamical systems using the self-organizing map. *IEEE Transactions on Neural Networks*, 15(5):1244–1259, 2004.

[6] 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*, 86(11):2240–2258, 1998.

[7] D. F. Specht. A general regression neural network. *IEEE Transactions on Neural Networks*, 2(6):568–576, 1991.

[8] A. Zaknich. Introduction to the modified probabilistic neural network for general signal processing applications. *IEEE Transactions on Signal Processing*, 46(7):1980–1990, 1998.

[9] 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*, 28(2):231–238, 1998.

[10] 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*, 32(1):17–19, 1996.

[11] 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*, pages 340–345, 2003.