Buletinul Ştiinţific al Universităţii "Politehnica" din Timişoara

Seria ELECTRONICĂ şi TELECOMUNICAŢII
TRANSACTIONS on ELECTRONICS and COMMUNICATIONS

Tom 49(63), Fascicola 1, 2004

Competitive learning methods for RBF neural network initializations - application to digital channel equalization

Nicolae Miclău[1]

Abstract – A complex-valued radial basis function neural network RBF is proposed for digital communications channel equalization. Performances are directly related to the clustering centers estimations. For this aim different competitive learning algorithms are developed. The network has complex centers and connection weights, but the nonlinearity of its hidden nodes remains a real-valued function. The RBF network is able to generate complicated nonlinear decision regions or to approximate an arbitrary nonlinear function in complex multidimensional space.

Keywords: Complex-valued radial basis function network RBF, Competitive learning.

I. INTRODUCTION

The channels used to transmit the data distort signals in both the amplitude and the phase, causing the intersymbol interference (ISI). Nonlinear active or passive devices causes further nonlinear distortions that affect the signals. The equalization methods consist in signal processing techniques used to restore the signals and recover their information. The best performance of all equalization techniques is obtained using the maximum likelihood strategies. The optimal solutions in the design of the nonlinear adaptive equalizers have been approached using the Bayesian decision theory.

It is known that problem of equalization may be treated as a problem of classification, so neural networks (NN) are quite promising candidates because they can produce arbitrarily complex decision regions. Studies performed during the last decade have established the superiority of neural equalizers comparative to the traditional equalizers, in conditions of high nonlinear distortions and rapidly varying signals [1].

On the other hand, the problems which severely restrict the practical implementation of multilayer perceptron (MLP) are the extreme length of training times, and unpredictable solutions. The learning difficulties lies in the fact that the error surface of an MLP is highly complicated and potential bad local minima exist for any gradient descent based algorithm.

RBF networks often provide a faster and more robust solution to the equalization problem. In addition, RBF neural network has a structure similar to the optimal Bayesian symbol decision equalizer.

Most available neural networks are real valued and are suitable for signal processing in real multidimensional space.

The signals with a variable envelope modulation, as for example the quadrature amplitude modulation (QAM) signals, are severely affected by the nonlinear distortion in phase and in amplitude. To compensate the distortions of QAM and phase shift keying (PSK) signals equalizers for complex signals are necessary because is preferred to preserve the concise formulation and elegant structure of complex signals [2].

In [3], [4] and [5] is proposed a complex radial basis functions (RBF) network, with one hidden layer. The inputs and outputs for this network are both complexes valued but the nonlinearity of hidden node is a real function.

Several learning algorithms have been proposed to update the RBF parameters. However, the most popular algorithm consists of an unsupervised learning rule for the centers of hidden neurons and a supervised learning rule for the weights of the output neurons. The centers are generally updated using the *k*-means clustering algorithm, orthogonal least squares (OLS) [3], or stochastic gradient algorithm (SG) [5]. The *k*-means algorithm has some potential problems: classification depend on the initials values of the centers of RBF, on the type of chosen distance, on the number of classes. If a center is inappropriate chosen it may never be updated, so it may never represent a class. OLS algorithm is a good technique, but is complicated to be used in practice. SG algorithm adapts all free parameters of the network simultaneously, but does not guarantee convergence to globally optimum network parameters.

In contrast with earlier work the proper choice of the centers is crucial for good performance.

[1] Facultatea de Electronică şi Telecomunicaţii, Departamentul Comunicaţii Bd. V. Pârvan Nr. 2, 300223 Timişoara, e-mail miclau@etc.utt.ro

Therefore, I proposed and tested in these paper two methods to update the RBF centers. First is a conventional competitive learning (CL), which can function as an adaptive method for clustering analysis problems encountered in statistical data analysis or unsupervised pattern recognition, also it can be used for vector quantization. Second is one improved CL, frequency sensitive competitive learning (FSCL) algorithm. FSCL introduces the strategy of reducing the winning rate of the frequent winners, strategy called conscience. The both algorithms work with complex numbers.

II. THE COMMUNICATION CHANNEL MODEL AND THE EQUALIZER ARHITECTURE BASED ON RBF NETWORK

A. *Communication channel model*

The general model of digital communication channel and the equalizer is depicted in Fig. 1. The source transmits a symbols sequence $x(t)$. The symbols belong to one certain alphabet and are assumed to be independent with an equal probability [1].

The linear part of communication channel is often represented by a finite impulse response (FIR) filter:

$$\tilde{y}(k) = \sum_{i=0}^{k-1} \theta_i x(k-i) \qquad (1)$$

The nonlinear part of the channel, which reflect the nonlinear distortion are taken into account by:

$$y(k) = g(\tilde{y}(k)) = g(x(k),\ldots,x(k-n+1),\Theta) \qquad (2)$$

where $g(.)$ is a nonlinear function and Θ is the vector of FIR coefficients:

$$\Theta = [\theta_1 \ldots \theta_{n-1}]^T \qquad (3)$$

At input of equalizer, observed signal is:

$$o(k) = g\left[\sum_{i=0}^{n-1} \theta_i x(k-i)\right] + b(k) \qquad (4)$$

If the signal x is a 4 QAM, the input constellation is given by:

$$x(k) = x_R + jx_I = \begin{cases} x^{(1)} = 1 + j \\ x^{(2)} = -1 + j \\ x^{(3)} = 1 - j \\ x^{(4)} = -1 - j \end{cases} \qquad (5)$$

The input symbols sequence $x(k) = [x(k)\ldots x(k-m-n+2)]$ is passed through the nonlinear communication channel model and produce at its output one vector $y(k) = [y(k)\ldots y(k-m+1)]^T$ with $n_e = 4^{n+m-1}$ distinct states, where m is he order of equalizer.

The channel output signal is affected by an additive noise $b(k)$, usually white Gaussian, and produces a corrupted signal $o(k)$.

The problem of equalization is to determine an estimation of the input signal using the received signal $o(k)$ and the desired delayed signal $x(k-d)$, where d is the delay of equalizer.

B. *The network architecture*

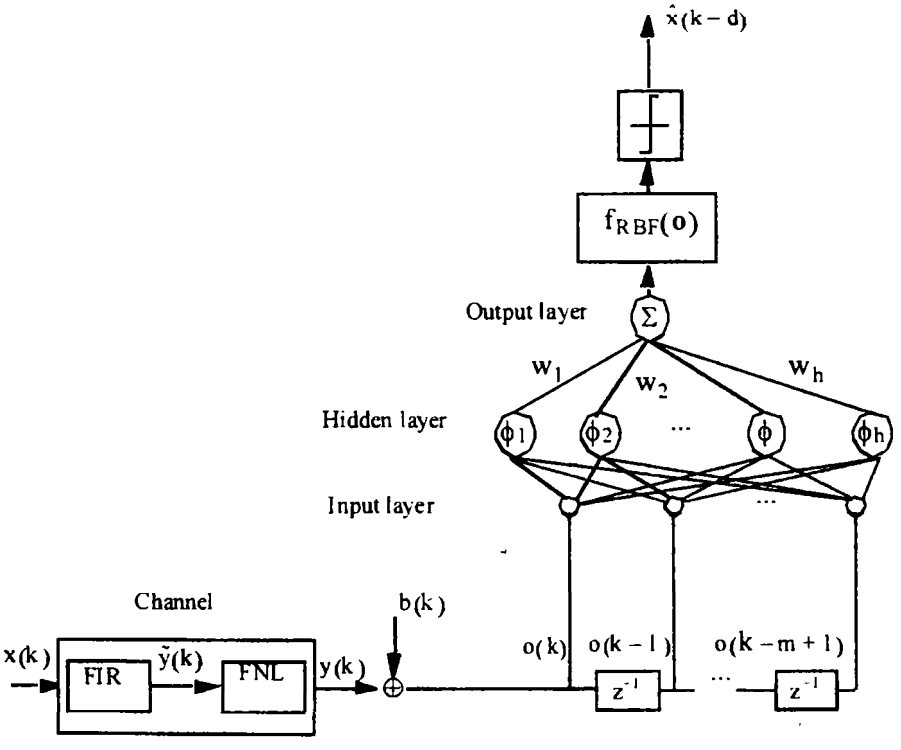

Fig. 1. Equalizer based on RBF neural network

From the NN point of view, the equalizer has to classify the received signal in one of the four possible classes $P_{m,d}$, according to the input signals:

$$P_{m,d} = \bigcup_{1 \le l \le 4} P_{m,d}^{(l)} \tag{6}$$

or:

$$P_{m,d}^{(l)} = \{y(k) | x(k-d) = x^{(l)}\}, 1 \le l \le 4 \tag{7}$$

The design of a RBF neural network may be viewed as a curve fitting approximation in a complex multidimensional space. According to this viewpoint, learning is equivalent to finding a surface in a multidimensional space that provides a best fit to the training data.

The construction of the RBF is depicted in Fig. 1. and involves three different layers. The input layer is made up of sensory units; second layer is a hidden layer which serves a different purpose that in a multilayer perceptron. The output layer supplies the response of the network to the activation patterns applied to the input layer.

The hidden layer is composed of an array of computing neurons, each having a parameter c_i vector called center. Each neuron computes a distance between its center and the network input vector. This distance may be of different types and it is subsequently divided by a parameter ρ_i, called width, which is the spread of the corresponding center. The result is passed through a real, nonlinear activation function. $\phi_i(\bullet, \rho_i)$

$$\phi_i = [\phi(o - c_i)^H (o - c_i), \rho_i], \quad 1 \le i \le n_h \tag{8}$$

where o is the complex input vector of n_h dimension, c_i is the centers vector of the radial basis functions, which is also a complex vector of n_h dimension, ρ_i is the center spread parameter, n_h is the number of computing nodes. The operator $(\bullet)^H = ((\bullet)^T)^*$, where $(\bullet)^T$ is the transposition operator and $(\bullet)^*$ is the complex conjugation operator.

The nonlinear output function is usually the Gaussian function:

$$\phi(\chi^2, \rho) = e^{-\frac{\chi^2}{\rho}} \tag{9}$$

The number of hidden neurons n_h is given by the number of possible states of the channel output n_e. A number n_h greater than n_e generates inutile computing. A number n_h smaller than n_e may degrade the performances of the network.

Similarity with the Bayesian equalizer imposes that the spread parameter $\rho = 2.\sigma^2$ where σ^2 is the noise dispersion given by relation:

$$\sigma^2 = E\|o(n) - c_i\|^2 \tag{10}$$

where E is the mean, the second order momentum.

The output layer of the network consists of eight neurons (two neurons, one for the real part and the other for the imaginary part of each class) with a linear function:

$$f_{RBF}(o) = \sum_{i=1}^{n_h} \phi_i w_i \tag{11}$$

where w_i are the complex weights. According to the relation (9) f_{RBF} becomes:

$$f_{RBF}(o) = \sum_{i=1}^{n_h} w_i . e^{-\frac{(o-c_i)^H (o-c_i)}{\rho_i}} \tag{12}$$

III. COMPETITIVE LEARNING ALGORITHMS

A. *Competitive learning standard algorithm*

The competitive learning (CL) standard algorithm calculates the distance between the input vector and the RBF centers vector. The distance may be of different types, usually the Euclidian norm is used [6]:

$$\|o(k) - c_i(k)\| = \sqrt{|o(k) - c_i(k)|^2 + \ldots + |o(k-m+1) - c_i(k-m+1)|^2} \tag{13}$$

The neuron j with a minimum distance is declared winner:

$$j = \arg\min \|o(k) - c_i(k)\|, \quad i = \overline{1, n_h} \tag{14}$$

The winning neuron center is moved with a fraction η towards the input.

The weights (centers) c_i, $i=1,n_h$, can be randomly chosen from the input vector o. The training algorithm iterates a number of times through the training data, adjusting the centers vector. The algorithm for updating the weight vector consists of the following steps [7]:

1. Pre-specify the number n_h of clusters and initialize the seed points $\{c_i, i=1,n_h\}$.
2. Given an input o, calculate the indicator function u_j by:

$$u_j = \begin{cases} 1, \text{if } j = c, \|o(k) - c_c(k)\|^2 = \min_i \|o(k) - c_i(k)\|^2 \\ 0, \text{otherwise} \end{cases} \tag{15}$$

3. Update the winning seed point c_i, by:

$$c_j^{new} = c_j^{old} + \eta u_j (o(k) - c_i(k)) \tag{16}$$

where η is a small positive learning rate.

The equation (15), (16) are called the Winner-Take-All rule.

The implementation of the above algorithm is quite easy; each weight (centers) vector is randomly

initialized, and then the step two and three are iterated until the iteration converges or freezes as the learning rate becomes zero or very small, or until the number of iterations reaches a pre-specified value.

B. *Frequency sensitive competitive learning*

The simple classical CL algorithm has the so called *dead unit* problem. For resolve this problem a notable improvement is the strategy of reducing the winning rate of the frequent winners. Each center keeps track of how many times it has won the competition. The notion used here is that if the center wins too often, "it feels guilty" and therefore pulls itself out of the competition [8]. The frequency sensitive competitive learning (FSCL) is an extension of CL, obtained by modifying (15) into the following:

$$u_j = \begin{cases} 1, \text{if } j = c, \gamma_c \|a(k) - c_c(k)\|^2 = \min_i \gamma_i \|a(k) - c_i(k)\|^2 \\ 0, \text{otherwise} \end{cases} \quad (17)$$

with the relative winning frequency γ_i of c_i defined as:

$$\gamma_i = \frac{s_i}{\sum_{j=1}^{nh} s_j} \quad (18)$$

where s_i is the winning times of c_i in the past. After selecting out the winner, FSCL then updates the winner only by Eq.(16) in the same way as CL, and meanwhile adjusting the corresponding n_c with:

$$n_c^{new} = n_c^{old} + 1 \quad (19)$$

FSCL can almost always successfully distribute n_h seed points into the input data set without dead-unit problem.

IV. SIMULATION RESULT

QAM input signals were generated using an uniform distribution, independently for the real part from the imaginary part. Simulations were done using the channel Chen's model [4]:

$$H(z) = (0.7409 - 0.7406i)(1 - (0.2 - 0.1i)z^{-1}) \cdot (1 - (0.6 - 0.3i)z^{-1}) \quad (20)$$

The order of FIR filter is chosen as n=3. The decision delay was chosen to be d=1. Thus, the output channel $y(k)$ will be of 64 possible states. A white noise $b(k)$ was generated with variance 0.125 and added to $y(k)$. The number of the hidden neurons was chosen n_h=64 and of the output neurons 8. The RBF centers were randomly initialized to a subset of channel output values. There were applied N= 6000 input signal sequences $x(k)$=[$x(k)$ $x(k$-1) $x(k$-2)] to train the equalizer.

The learning rate for the CL algorithm , and FSCL algorithm is η=0.09, and the decay of the learning was chosen1-1/(N).

First we use CL algorithm to find the centres of RBF network. In Fig. 2 are represented the output channel states y(n), the corrupted received signal o(n), the initial and final positions of the RBF centers in case of noise variance 0.125 and after one epoch of training. Each of the center vectors moved to a cluster

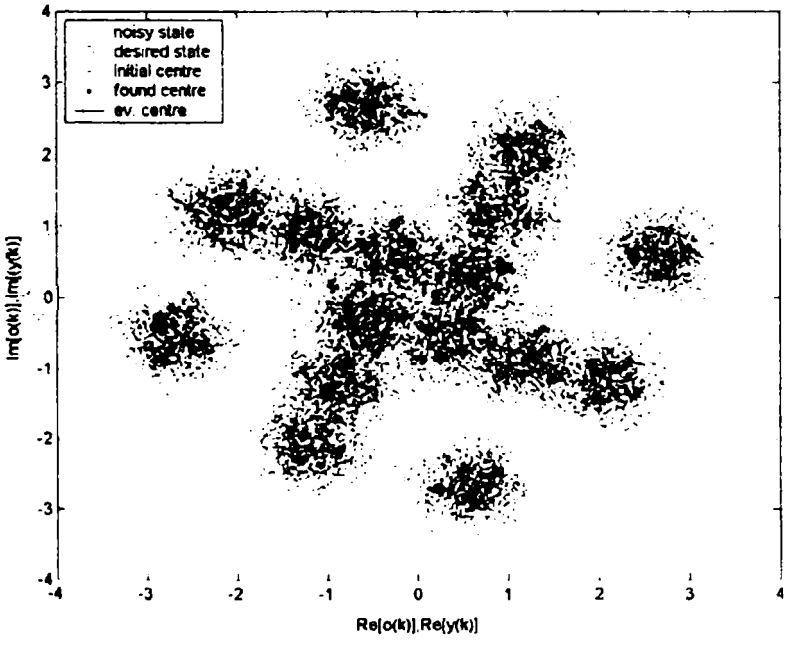

Fig. 2. Bidimensional representation of output channel states without noise , inputs of RBF network, initial and final positions of the RBF centers for N= 6000 sequences after one epoch of training with CL algorithm.

center regardless of the initial values of these center vectors.

To illustrate the *dead unit* problem of CL algorithm, I initialize the centers vectors by random complex numbers in the interval between 4.5 and 5.5. In Fig. 3 is shown the evolution of the centers; is evident that exist only one center which move and 63 *dead units*.

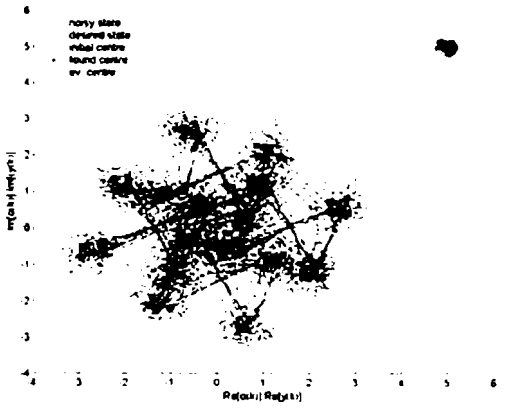

Fig. 3. Other initialization of the centers. The learning traces (trajectories of center vectors) during the learning process, obtained by the classical CL algorithm after 5 epochs of training

In this case the performance of RBF network decrease dramatically and the CL algorithm is inefficient.

The solution for this problem is the FSCL algorithm. In Fig. 4 is represented the result using the *conscience* strategy.

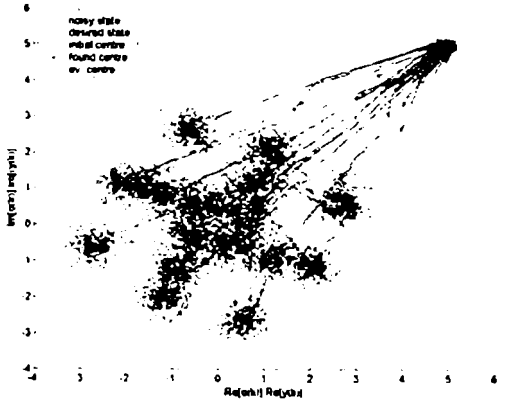

Fig. 4. The learning traces obtained by FSCL algorithm after 15 epochs of training

In contrast with CL algorithm, an important number of the centers are moved.

To update the output neurons weights a supervised algorithm is used: the least mean square (LMS) algorithm.

LMS minimizes the mean square error:

$$MSE = \frac{1}{N}\sum_{i=1}^{N} e_i(k)^2 \qquad (21)$$

where N is the number of input sequences and the complex error $e(k)$ is determined with:

$$e(k) = x(k-d) - f_{RBF}(o) \qquad (22)$$

For each sequence it has been calculated the error $e(k)$ and than the MSE. The output weights were trained according, in order to minimize the MSE.

Fig.5 depicts the MSE evolution during 2000 epochs for both algorithms. For the CL algorithm the vector centers was initialized randomly from the input data, closed to each cluster to avoid the *dead unit* problem. For the FSCL the centers were initialized by random complex numbers in the interval between 4.5 and 5.5.

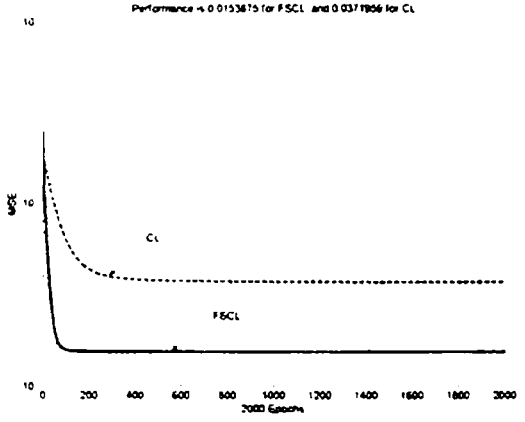

Fig. 5. Evolution of mean square error during 2000 iterations for CL, and FSCL algorithms.

We can observe that FSCL algorithm is better than CL even if CL is utilized in the best situation, with initial centers vector initialized closed to the each cluster of data.

Fig. 6 represents the partition signals space for a delay of d=1, for the obtained with FSCL algorithm.

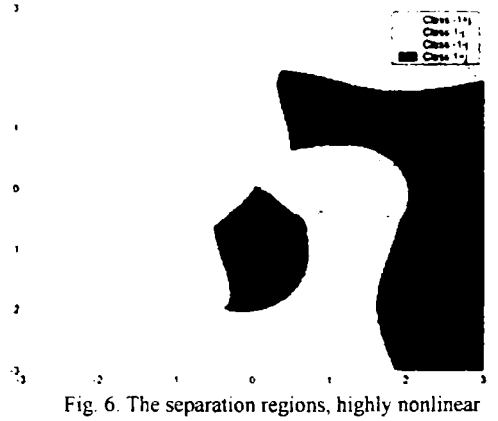

Fig. 6. The separation regions, highly nonlinear

V. CONCLUSION

The proper choice of the centers is crucial for a good performance of the complex RBF network.

For these aim two algorithms was developed in a complex form. The both algorithms have a fast convergence, and the implementation is quite easy.

The CL algorithm has a dead unit problem. In order to avoid this problem a good initialization of the centers (near of the input data) is necessarily. Other solution is the FSCL algorithm which resolves this problem.

The FSCL algorithm is superior, even if CL algorithm was initialized in a good manner.

The performance of the both algorithm may be increased in two ways:

- increasing the number of the input data;
- increasing the number of the training epochs;

So this algorithms is adequate to the adaptive equalization of fast varying signals corrupted with strong linear and nonlinear distorsions.

VI. ACKNOWLEDGMETS

The author is greatly indebted to Prof. dr. ing. Miranda Nafornita for stimulating discussions.

REFERENCES

[1] Safwan EL Assad, Nicolae Miclau et Denis Hamad, "Réseaux neuronaux RBF utilisés en égalisation adaptative des canaux de transmission non linéaires", *ICISP2-2003*, Agadir, Maroc, Juin 2003.

[2] Nicolae Miclau, Corina Botoca, Georgeta Budura, "Nonlinear Complex Channel Equalization Using A Radial Basis Function Neural Network", *Neurel Proc.*, Beograd, Yugoslavia, August 2004.

[3] S. Chen, S. McLaughlin, B. Mulgrew, "Complex-Valued Radial Basis Function Network. Network Architecture And Learning Algorithms", *Signal Processing*, No.35, pp. 19-31. 1994

[4] S. Chen, et al., "Complex-Valued Radial Basis Function Network. Part II: Application To Digital Communications Channel Equalization", *Signal Processing*. No. 36, pp. 175-188,1994

[5] I. Cha , S. Kassam, "Channel Equalization Using Complex-Valued Radial Basis Function Networks", *IEEE Journal Select. Areas Commun.*, vol13, pp.122-131, Jan, 1995

[6] D. Hamad, S. El Assad et J. Postaire, "Algorithmes d'apprentissage compétitif pour la classification automatique", *TISVA'98*, Oujda, Maroc, April, 1998

[7] LeiXu, A. Kryzak. E.Oja "Rival Penalized Competitive Learning for Clustering Analysis, RBF Net, and Curve Detection", *IEEE Trans. On Neural Networks*, Vol. 4, No. 4, 1993

[8] S. HAYKIN, "*Adaptive filter theory*", Prentice Hall, Third edition, 1996.