SIMULATION OF GAUSSIAN POTENTIAL FUNCTION NETWORK FOR ADAPTIVE EQUALIZATION

Cüneyd Fırat

Software Engineering and Simulation Group of
Information Technologies Research Institute
TUBITAK-MRC
Gebze-Kocaeli/Turkey

ABSTRACT

This study consists of simulation of Gaussian Potential Function Network (GPFN) [1] and application to filtering problem called intersymbol interference (ISI) equalization which is conventionally handled with digital equalizers. A GPFN is a Radial Basis Function (RBF) like net. It has three layers; one input layer, one hidden layer, which includes the nodes with gaussian activation function, and the last is the output layer. Its distinct feature comes from its ability of automatic recruitment of hidden layer nodes. The filtering of signals, which are corrupted with additive white gaussian noise (AWGN) to represent ISI, performed with this simulation showed that this network is a suitable tool for adaptive equalization.

1.INTRODUCTION

Many digital communication channels are subject to intersymbol interference (ISI). This interference is usually a result of the restricted bandwidth allocated to the channel and/or the presence of multipath distortion in the medium through which the information is transmitted. In symbol-by-symbol equalization, the problem is that of using the information in the observed channel outputs to produce an estimate of the current output. The device or algorithm that performs this function is the equalizer. Equalizers have different architectures and algorithms like transversal equalizer or maximum likelihood estimator which has theoretically the best performance but needs extremely expensive computation.

Neural net models are classified by net topology, node characteristic and training or learning rules. These rules arranges the way the net learns the required assignment. It includes the initialization and adaptation of parameters and weights for minimum error. The simple node in a neural network model called perceptron. The capabilities of single perceptron are limited to linear decision boundaries and simple logic functions. However, by cascading peceptron in layers, called multilayer perceptron (MLP), we can implement complex decision boundaries and arbitrary boolean expressions.

Initial work [2]demonstrated that MLP equalizers were superior to conventional transversal and decision feedback equalizers in terms of the usual measure of equalizer performance, which is bit errror rate (BER). On the other hand, the work also highlighted several of the difficulties that are well known in the wider application of MLP's. These are extreme length of training times; the indeterminate nature of the training times; the lack of metedology for architecture selection. These problems are largely unsolved and severely restricted the practical application of MLP's in this area.

An important MLP model is the Radial Basis Function network (RBF). RBF network has close relationship to Bayesian methods for channel equalization and interference rejection. RBF is a two layer network whose output nodes form linear combination of the basis functions computed by the hidden layer nodes. The basis functions in the hidden layer produce a significant nonzero response only when the input falls within a small localized region of the input space. Although implementation vary, the most common basis is a 'Gaussian Kernel' function. The node outputs are in the range from zero to one so that the closer the input to the center of the gaussian the larger the response of the node. GPFN is an RBF like network, which differs in learning scheme called hierarchically self organizing learning scheme. The nodes in hidden layer of GPFN is created automatically based on some checkings. This property of the network also provides the network with not sticking to local minimums.

This study consists of replacing Linear Transversal Digital Equalizer (LTE) with GPFN for estimating the digital signals corrupted by additive white gaussian noise (AWGN) and measuring and comparing the performance based on BER.

2.SUPERVISED LEARNING WITH GAUSSIAN POTENTIALS

2.1 Gaussian Potential Function Network

A potential-function network approximately realizes a "many-to-one" continuous mapping by synthesizing a potential field over the input domain by a number of potential functions. The potential field ϕ can be described as the weighted summation of a

number of potential functions, $\psi(x, p_i), i = 1, \ldots, M$ characterized by parameter vector, p_i, $i=1,\ldots,M$

$$\phi(x) = \sum_{i}^{M} c_i \psi(x, p_i) \tag{1}$$

where M, c_i, and p_i respectively, represents the number of potential unit functions (PFU), the summation weight, and parameter vector of the i^{th} potential function.

The unnormalized gaussian function is selected for the construction of potential-function network (PFN) since the function is highly nonlinear but has many well defined features, owing to its use in probability theory. A gaussian potential function is defined by;

$$\psi_i = \psi(x, p_i) = e^{-\frac{d(x,p_i)}{2}} \tag{2}$$
$$d(x, p_i) = d(x, m^i, K^i) = (x - m^i)^t K^i (x - m^i)$$

where x represents an input pattern and m^i, and K^i represents respectively the mean vector and the shape matrix (defined by the inverse of the covariance matrix) of the i^{th} potential function.

We can write $d(x, m^i, K^i)$ in an expanded form;

$$d(x, m^i, K^i) = \sum_j \sum_k k^i_{jk} (x_j - m^i_j)(x_k - m^i_k) \tag{3}$$

where

x_j : j^{th} element of input vector x

m^i_j : j^{th} element of mean vector m^i

k^i_{jk} : $(j,k)^{th}$ element of k^i matrix

and k^i_{jk} can be written in terms of standard deviation σ^i_j, σ^i_k and correlation coefficient h^i_{jk} as follows;

$$k^i_{jk} = \frac{h^i_{jk}}{\sigma^i_j \sigma^i_k} \tag{4}$$

Instead of using the general form of k^i_{jk}, an approximation is possible in the following form

$$k^i_{jk} = \begin{cases} \frac{1}{\sigma_i^2} & \text{if } j = k \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

In general, the input vector is multidimensional. The existence of the shape matrix provides the possibility of having different variances in different dimensions i.e. different crossections of GPFs are possible. By this approximation some flexibility is lost that will be compensated using more Gaussian Potential Function Units (GPFU).

The network model proposed here is composed of three layers; the input layer, the hidden layer, and the output layer. The input and output layers are composed of linear units, whereas the hidden layer consists of nonlinear GPFUs, which produce gaussian potential functions. The weighted output values of the GPFU's are summed by the connections between the hidden layer and the output layer in order to synthesize the required potential fields. The three layered PFN with the GPFU's configured at the hidden layer is called the GPFN. Figure 1.(a) illustrates the schematic diagram of a GPFN. Figure 1(b) shows a detailed structure of the i^{th} GPFU. The calculation of (3) starts with the subtraction of the mean vector of the i^{th} GPFU from the input vector at the subtraction nodes.

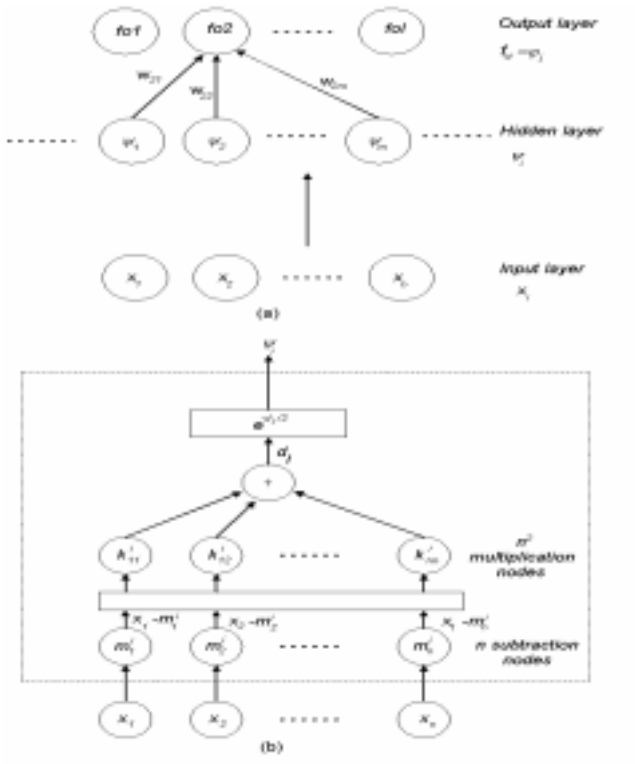

Figure 1. Schematic diagram of GPFN

Then the components of the vector obtained at the subtraction nodes are cross correlated among themselves by the cross-correlator to obtain N^2 cross-correlated term, is multiplied by the corresponding k^i_{jk} of the shape matrix K^i at the multiplication nodes and summed for d_i. The output of the GPFU is then generated by exponentiating d_i.

2.2 Learning in GPFN

Learning in GPFN consists of determining minimum necessary number of GPFU's and the adjustment of the mean vector and the shape matrix as well as the summation weights. The distinct feature of learning for GPFN is its capability of automatically

(based on some decisions) recruiting of necessary computational units (nodes).

The automatic recruitment of computational units is based on the following decisions;

- Whether a new teaching pattern should be accommodated by the GPFU's already generated from the past training or by a new GPFU.
- Whether the accommodation boundaries of individual GPFU's, defined by the contour of the equidistances from the center of Gaussian potential functions, should be readjusted to ensure error convergence as well as fast learning.

This learning scheme is called **Hierarchically Self-Organizing Learning Scheme (HSLS)**. This scheme has the following implications;

- It starts with learning global mapping features based on a small number of computational units with larger size of accommodation boundaries, then proceeds to learning finer mapping details by increasing the number of computational units and reducing the size of accommodation boundaries.
- It changes the dimensions and shape of the error surface of the error function defined in terms of the network parameters, when the performance of error convergence is considerably degraded. This helps to avoid to sticking on a flat or very mildly sloping surface which might cause trouble.

The learning algorithm is composed of two parts; the first is concerned with the adjustment of the network parameters and the second with the decision on the minimum necessary number of GPFU's

The parameter learning is conducted with gradient-descent method in which the parameters are updated according to the error function which is the function of network parameters.

Since the gradient-descent method is well known in general, the decision part of HSLS will be explained here.

2.1.1 Hierarchically Self-Organizing Learning

This scheme performs automatic recruitment of new GPFUs and readjustment of the accommodation boundaries of every GPFU. The network will start with small number of GPFUs which have large accommodation boundaries for rough but global mapping. This will give bad performance. But as the new GPFUs are generated and accommodation boundaries are reduced, the input space will be shared out by more GPFUs giving more precise approximation.

For the accommodation boundary an effective radius r_i is defined in the form of a hypersphere , H_i, defined in the input space

$$H_i(r_i) = \left\{ x | d(x, m^i, K^i) \le r_i^2 \right\} \tag{6}$$

In addition a GPFU is assigned a property to represent the particular class of patterns. This assignment is done at the time a GPFU is generated by taking the same class represented by the teaching pattern that invoked its generation.

Accomodation generation rules are as follows :

If the currently introduced teaching pattern falls inside the boundary and belongs to the same class of any GPFU then generation of a new GPFU is not required. Present parameters will be updated. A teaching pattern will be accepted in the boundary of a GPFU by the condition

$$e^{-\frac{d(x,m^i,K^i)}{2}} \ge e^{-\frac{r_i^2}{2}} \tag{7}$$

The effective radii of GPFUs will be reduced for precise learning. The rate of reduction is important. Fast reduction will cause generation of more than necessary number of GPFUs, because individual GPFUs can not find enough time to converge their optimum shapes. On the other hand, slow rate will certainly result in minimum number of GPFUs, but will need more learning cycle (random presentation of teaching patterns).

The best way to reduce r_i is to measure the progress of the learning. Since it supplies more GPFU and better learning it should be reduced when learning progress is saturated with currently available GPFUs. The progress of learning can be measured in two ways:

- With performance index P defined as:

$$P \equiv e^{-E_{rms}} \tag{8}$$

where E_{rms} represents the root-mean-square error for N teaching patterns

$$E_{rms} = \sqrt{\frac{2}{MN} \sum_{p=1}^{N} E_p} \tag{9}$$

- With parameter saturation vector s_j, defined for the jth output unit as:

$$s_i^{new} = \alpha \frac{\partial E_p}{\partial n_i} + (1-\alpha) s_i^{old} \tag{10}$$

where σ is a positive constant between 0 and 1.

The purpose of defining the parameter saturation vector is to monitor $\frac{\partial E_p}{\partial n_j}$, since the saturation of the performance on learning implies the convergence of the parameters to an extremum achievable with the number of currently available

GPFUs. In Equation (10) s_j moves toward $\frac{\partial E_p}{\partial n_j}$. Two cases can be identified as

- If the network parameters remain stationary at a point or wonder around a point in the parameter space, the norm s_j defined below gradually decreases toward zero.
- If the network parameters are on the way to converge to a point in the parameter space along a certain direction, the magnitude of s_j gradually increases or decreases toward $\frac{\partial E_p}{\partial n_j}$

Therefore, by monitoring the value of $\|s_j\|$ (the maximum of the sum of the columns), the saturation of the network performance on learning can be detected.

3.EQUALIZATION WITH GPFN

The simulation was performed based on the model shown in Figure 2. The binary symbols that have been passed through a linear dispersive channel and then corrupted by additive white gaussian were detected provided that LTE replaced with GPFN (Figure 3). A linear dispersive channel introduces intersymbol interference (ISI) and can be modeled as a finite impulse response filter (FIR):

$$C(z) = \sum_{i=0}^{l_c-1} a_i z^{-1} \tag{11}$$

where l_c the length of the channel impulse response and a_i 's are the channel coefficients. The received signal $r(k)$ has white gaussian noise in addition to channel output is;

$$r(k) = \sum_{i=0}^{l_c-1} a_i x(k-i) + w(k) \tag{12}$$

The input $x(k)$ is chosen from {-1,1} with equal probability and assumed to be independent. The channel output is corrupted with additive noise of zero mean and variance σ^2_n. At time k, the detector (or equalizer) provides an estimate of $x(k-d)$ as $\hat{x}(k-d)$ based on finite number of noisy observed samples, $r(k)$, $r(k-1)$, . . ., $r(k-N+1)$, where d is the decision delay and N is the number of taps. The goal is to minimize the probability of bit error.

The channel coefficients were chosen as $a_0=1$ and $a_1=0.5$. The received signals r_k, r_{k-1}, r_{k-2}, where $N=3$, are the inputs of the GPFN. GPFN is assigned to decide which signal (1 or –1) is transmitted. If the output of the GPFN has different sign with the actual transmitted signal, it is evaluated as an error.

The network generated 2 or 3 GPFU's at different SNR levels (e.g. 3 GPFU's at 7dB). At high noise levels the error surface is very serrated. On the other hand, the network is very sensitive to parameters since the hidden layer nodes are automatically generated. So selection of parameters play very important role in succession of the network.

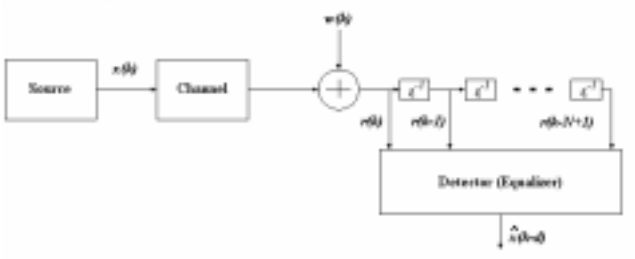

Figure 2. System model used

The network was tested at several SNR levels starting from 7 dB to 16 dB with 1 dB increment. It was trained using 5000 symbols and the bit error performance was evaluated over 20000 symbols. Results which compares the LTE and GPFN both trained and tested on the same conditions is in Figure 4, which shows the *log(tot. number of bit errors/20000)* vs. corresponding *SNR*. Both filters reached the 0 error performance at 16 dB.

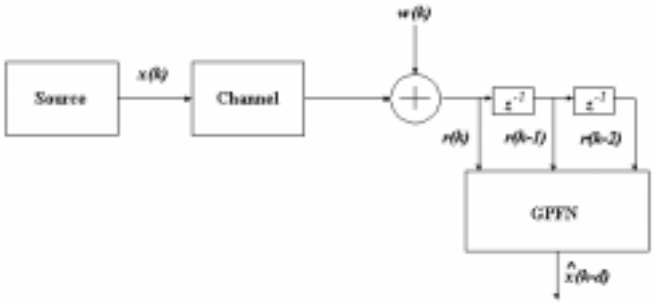

Figure 3. The GPFN used instead of Equalizer (LTE)

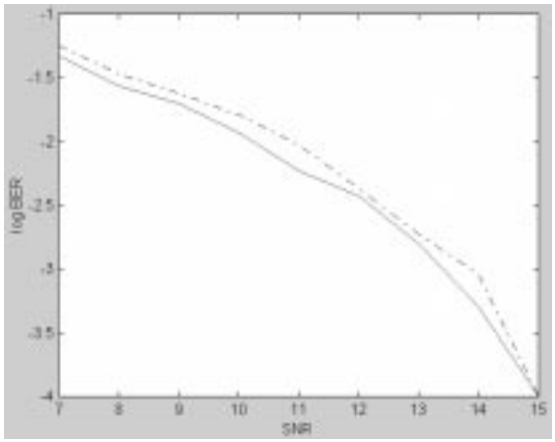

Figure 4. Log(bit error (BER)/20000) vs corresponding SNR (dB). '-.' line represents GPFN curve whereas '-' represents LTE

4.CONCLUSION

In many applications of neural networks there may not be universally recognized “best” or “optimal” solution. The network is presented with training set of input/output pairs to learn a nonlinear mapping from input to output. Then the network tested on unseen data, and if performance goals are met, the operation is deemed a success.

GPFN and Hierarchically Self Organizing Learning Scheme (HSLS)provides a base for adaptive structures. So this scheme was used for filtering ISI in linear dispersive channels instead of conventional mechanism called linear transversal digital equalizer (LTE). When the results of this study concerned, GPFN and HSLS performed good in general but not better than LTE. But both curves are very close to each other. Although this might be related to the improper selection of several parameters at the beginning of learning, there is no definite way of selecting these parameters for different application. Because, it is strongly dependent on the nature of the application. So it needs some experimental information. Also selection of the neural network architecture is a very effective factor and it is obvious that GPFN and HSLS does not leave the selection of hidden layer nodes to the user. So an effective optimization of architecture should be implied to the network at the time of constant parameters selection like learning rate or saturation vector parameters, etc..

In the light of these facts, it might be said that GPFN is apt to succeed a good performance for adaptive equalization but needs more experimental study

5.REFERENCES

[1] Kosko, B., “Supervised Learning With Gaussian Potentials”, *Prentice Hall,* pp.189-207,1992

[2] Siu, S., Gibson, GJ., Cowan, C.F.N., “ Decision feedback equalization using neural network structures and performance comparison with the standard architecture” *IEE Proceedings Part I,* volume 137, no. 4, pp. 221-225, August 1990

[3] Mulgrew, B., “Applying radial basis functions”, *IEEE Signal Processing Magazine*, pp.50-65, March 1996

[4] Lippmann, R. P., “An Introduction to Computing With Neural Networks”, *IEEE ASSP Magazine*, April 1987

[5] Hush, D. R., Horn, B. G., “Progress in Supervised Neural Networks; What is New Since Lipmann ?”, *IEEE Signal Processing Magazine,* January 1993

[6] Alexander, I., Morton, H., “An Introduction to Neural Computing” *Chapman & Hall,* pp. 20-22, 1991