Adaptive Noise Filtering of 16-Ary QAM Signal Using Multiplicative Neural Network

Shivendra Kumar Tiwari[1], Ms. Priya Shivhare[2]

[2]*Asst.Prof. IFIT, Bhopal*

Abstract - In this paper equalization of higher level modulation scheme such as 16-Ary QAM is done with the help of multiplicative neural network. Equalization is used at the receiver to combat noise. Performance comparison with 4- Ary QAM is also shown to provide significance of Multiplicative neural network.

Keywords - Channel equalization, 4-QAM signal, multiplicative neuron, feed forward neural network.

1. INTRODUCTION

In today's world, digital transmission has a tremendous impact on the human civilization. There has been a sea change in modern day living and the credit goes to the development in digital communication technology. With expanding communication networks, as we move towards a more information centric world and the demand for very high speed efficient data transmission over physical communication channels increases, communication system engineers face ever increasing challenges in utilizing the available bandwidth more efficiently so that new services can be handled in a flexible way. The objective of any digital communication system is to convey information, with the minimum possible introduction of error. Some typical forms of transmission media are open-wire lines, coaxial cables, microwave radio, optical fibers, satellite links etc. These media differ, essentially, in the volume of data per unit time which they can transmit. This data rate is limited by the noise and distortion introduced in the communication channel.

The distortions (phase-delay variations) introduced in the communication channel cause the transmitted symbols to spread and overlap over successive time intervals, resulting in a phenomenon, known as Inter Symbol Interference (ISI). In addition to ISI, the transmitted symbols are subjected to other impairments such as thermal noise, impulse noise and non-linear distortions arising from the modulation and demodulation process, cross talk interference, the use of amplifiers and converters etc. All the signal processing techniques used at the receiver end to combat distortions introduced due to channel impairments and recover the transmitted symbols accurately, are referred to as equalization schemes.

A common means of overcoming this problem has thus been to introduce an inverse filter into the receiver to equalize the channel. This simplistic view offers a satisfactory solution provided the channel transfer function is known and its inverse is convergent. However, such an inverse will be unstable in non-minimum phase channels (zeros outside the unit circle in the z-plane). Further, the designer does suffer from the disadvantage of having no a-priori knowledge of the channel transfer function and this function will be time-varying when the channel conditions are not stationary. For these reasons, in realistic situations equalizers are commonly adaptive in nature; that is, they automatically adjust their parameters when subjected to some external stimuli. Adaptive equalizers are characterized in general by their structures, the learning algorithms and the use of training sequences. Bandwidth efficient data communication requires the use of adaptive equalizers. Adaptive equalization algorithms need to exhibit some form of learning and this learning property is naturally found in artificial neural networks. Hence, many avenues have been established in the application of neural networks to adaptive equalization problems.

2. ADAPTIVE NOISE FILTERING

The block diagram of adaptive noise filtering using neural network in Fig.1 is described as follows [8]. The external time dependant inputs consist of the sum of the desired signal d(k),and the interfering additive white Gaussian noise v(k).

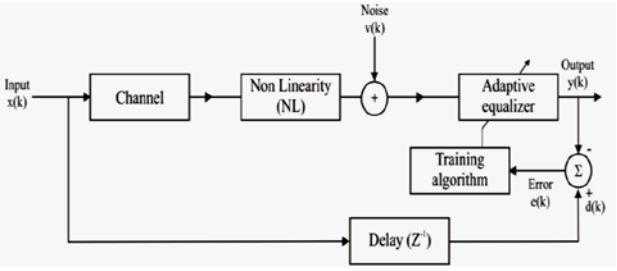


Fig.1 Adaptive noise filtering using neural network

The adaptive filter has a finite impulse response (FIR) structure. The impulse response is equal to the filter coefficients. The coefficients for a filter of order p are defined as

$$w_k = [w_k(0), w_k(1), \ldots\ldots, w_k(p)]^T \quad (1)$$

A predefined delayed version of the original signal forms the training sequence to provide reference points for the adaptation process. The criterion for optimization is a cost

function or the error signal which is the difference between the desired and the estimated signal given by

$$e(k) = d(k) - y(k) \quad (2)$$

The desired signal is estimated by convolving the input signal with the impulse response expressed as

$$d(k) = w_k^T \quad (3)$$

Where, $x(k) = [x(k), x(k-1), \ldots\ldots, x(k-p)]^T$ is the input signal vector. The filter coefficients are updated at every time instant as

$$w_{k+1} = w_k + \Delta w_k \quad \ldots..(4)$$

Δw_k is a correction factor for the filter coefficients.

The block diagram of a channel equalizer using MNN is shown in Fig.1. The transmitter sends a known training sequence to the receiver. A sequence of 1000, equiprobable, 16-QAM complex valued symbol set, in which the input signal takes one of 4 different values given by all possible combinations of $\{\pm 1, \pm 3\} + j \times \{\pm 1, \pm 3\}$,where $j = sqrt(-1)$ is generated. In the absence of the noise the output signal occupies well-defined M states of the M-Ary QAM signal constellation. When the signal is passed through the AWGN, it becomes a stochastic random process. Decision boundaries can be formed in the observed pattern space to classify the observed vectors between 4 classes. For equalization, the adaptive filter is used in series with the unknown system on the test signal by minimizing the squared difference between the adaptive equalizer output and the delayed test signal. The task of the equalizer is to set its coefficients in such a way that the output is a close estimate of the desired output. Depending on the value of the channel output vector, the equalizer tries to estimate an output, which is close to one of the transmitted values. The neural equalizer separately processes the real and imaginary part using the multiplicative, split complex, neural network model. The split complex approach is generally used to avoid singular points and critical selection of network parameters like the weights, bias, learning rate and momentum factor.

3. BACK PROPAGATION ALGORITHM

An error back propagation (BP) based learning using a norm-squared error function is described below [8]. The algorithm is first developed for single hidden layer network which is then extended to multi layer NN architecture. The aggregation function is considered as a product of linear functions in different dimensions of space. Fig.2 shows the structure of a multiplicative neuron. This kind of neuron itself looks complex in the first instance but when used to solve a complicated problem needs less number of parameters as compared to the existing conventional models.

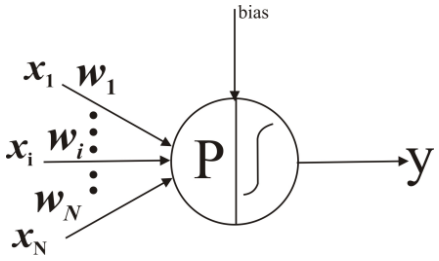


Fig.2 Structure of a single multiplicative neuron

Here the operator is a multiplicative operation as given in equation 3. The aggregation u before applying activation function is given by:

$$u = \prod_{i=1}^{n}(w_i x_i + b_i) \quad ..(5)$$

The output at the node y is given by

$$y = f(u) = \frac{1-e^{-u}}{1+e^{-u}} \quad ..(6)$$

The mean square error is given by

$$E = \frac{1}{2N}\sum_{p=1}^{N}(y^p - y_d^p)^2 \quad ..(7)$$

Where, p is the number of input patterns

The weight update equation for the split complex back propagation algorithm is given by

$$\Delta w_i = -\eta \frac{dE}{dw_i} \quad ..(9)$$

$$= -\frac{1}{2}\eta(y-d)(1+y)(1-y)\frac{u}{(w_i x_i + b_i)}x_i \quad (9)$$

Where η the learning rate and d is the desired signal. The bias is updated as

$$\Delta b_i = -\eta \frac{dE}{db_i} \quad ..(10)$$

$$= -\frac{1}{2}\eta(y-d)(1+y)(1-y)\frac{u}{(w_i x_i + b_i)} \quad (11)$$

$$w_i^{new} = w_i^{old} + \Delta w_i \quad (12)$$

$$b_i^{new} = b_i^{old} + \Delta b_i \quad (13)$$

The weights are updated after the entire training sequence has been presented to the network once. This is called learning by epoch. The algorithm [8] is extended to train multi layer multiplicative feed forward neural network as follows.

The symbols used are:

No is the number of inputs in the input layer.

n is the number of hidden layers in the FF network.

Nn is the number of neurons in the n^{th} hidden layer.

K is the number of outputs in the output layer.

j_n is the j^{th} neuron of the n^{th} hidden layer.

y_{jn} is the output of the j^{th} neuron of the n^{th} hidden layer.

y_{dk} is the desired output of the k^{th} neuron in the output layer.

y_k is the actual output of the k^{th} neuron in the output layer.

w_{jnjn-1} is the weight of the connection between j^{th} neuron of the $(n-1)^{th}$ layer and the j^{th} neuron of the n^{th} layer.

b_{jnjn-1} is the bias of the connection between j^{th} neuron of the $(n-1)^{th}$ layer and the

j^{th} neuron of the n^{th} layer. The output of the j^{th} neuron in the first hidden layer is given as

$$y_{j1}^{1}=\left\{\prod_{j0=1}^{N0}\left(w_{j1j0}+b_{j1j0}\right)\right\} \quad (14)$$

for j_1=1,2,….,N1 and represents j^thinput in the input layer and f(.) is the activation function defined by

$$f(y)=\frac{1-e^{-y}}{1+e^{-y}} \quad (15)$$

The output of the j^thneuron in the second hidden layer is given as

$$y_{j2}^{2}=f\left\{\prod_{j1=1}^{N1}\left(w_{j2j1}y_{j1}^{1}+b_{j2j1}\right)\right\} \text{ for } j_2=1,2,\ldots\ldots,N_2 \quad (16)$$

The output of the jth neuron in the nth hidden layer is given as:

$$y_{jn}^{n}=f\left\{\prod_{jn-1=1}^{Nn-1}\left(w_{jnjn-1}y_{jn-1}^{n-1}+b_{jnjn-1}\right)\right\} \text{ for } j_n=1,2,\ldots\ldots,N_n \quad (17)$$

The output of the k^{th} neuron in the output layer is given as

$$y_k=f\left\{\prod_{jn=1}^{Nn-1}\left(w_{kjn}\,y_{jn}^{n}+b_{kjn}\right)\right\} \text{ for } k=1,2,\ldots\ldots,k \quad (18)$$

A simple gradient descent rule, using a mean square error function is used for computation of weight update.

$$E_{MSE}=\frac{1}{2PK}\sum_{k=1}^{K}\sum_{p=1}^{P}\left(y_{dk}^{p}-y_{k}^{p}\right) \quad (19)$$

Where y_k^p and y_{dk}^p are the actual and desired values, respectively, of the output of the k^{th} neuron for the p^{th} pattern in the output layer. P is the number of training patterns in the input space. The weights are updated as below. Weights between output layer and the n^{th} hidden layer are given by:

$$\Delta w_{kjn}=-\eta\frac{dE_{MSE}}{dw_{kjn}} \quad (20)$$

$$=\eta d_k\frac{\left\{\prod_{jn=1}^{Nn-1}\left(w_{kjn}\,y_{jn}^{n}+b_{kjn}\right)\right\}}{\left(w_{kjn}\,y_{jn}^{n}+\mathbf{b_{kjn}}\right)}\mathbf{y_{jn}^{n}} \quad (21)$$

$$\mathbf{d_k}=\frac{1}{PK}\left\{\sum_{k=1}^{K}\sum_{p=1}^{P}(y_{dk}^{p}-y_{k}^{p})\left\{\frac{1}{2}(1+y_{k}^{p})(1-y_{k}^{p})\right\}\right\} \quad (22)$$

$$\Delta\mathbf{b_{kjn}}=\mathbf{d_k}\frac{\left\{\prod_{jn=1}^{Nn-1}\left(w_{kjn}y_{jn}^{n}+b_{kjn}\right)\right\}}{\left(w_{kjn}y_{jn}^{n}+b_{kjn}\right)} \quad (23)$$

$$=\frac{\Delta w_{kjn}}{y_{jn}^{n}}$$

Weights between n^{th} and $(n-1)^{th}$ hidden layer

$$\Delta w_{jnjn-1}=-\eta\frac{dE_{MSE}}{dw_{jnjn-1}} \quad (24)$$

$$=\eta d_k\frac{\left\{\prod_{jn=1}^{Nn}\left(w_{kjn}y_{jn}^{n}+b_{kjn}\right)\right\}}{\left(w_{kjn}y_{jn}^{n}+b_{kjn}\right)}\,w_{kjn}\,\frac{dy_{jn}^{n}}{w_{jnjn-1}}$$

$$=\Delta b_{jnjn-1}=\frac{\Delta w_{jnjn-1}}{y_{jn-1}^{n-1}} \quad (25)$$

Similarly, we can write equations for weight change between the hidden layer 1 and the input layer. The weights and biases are updated as

$$w_i^{new}=w_i^{old}+\Delta w_i \quad (26)$$

$$b_i^{new}=b_i^{old}+\Delta b_i \quad (27)$$

4. PROPOSED METHODOLOGY

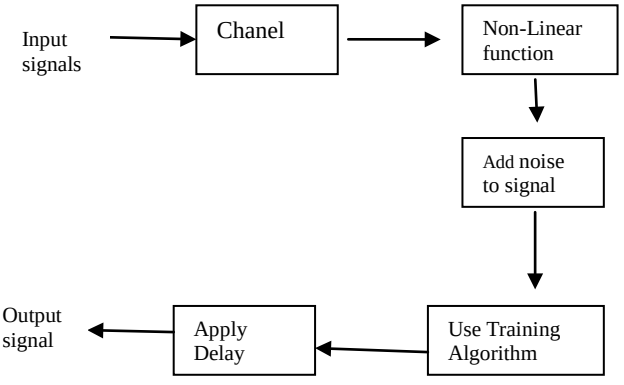
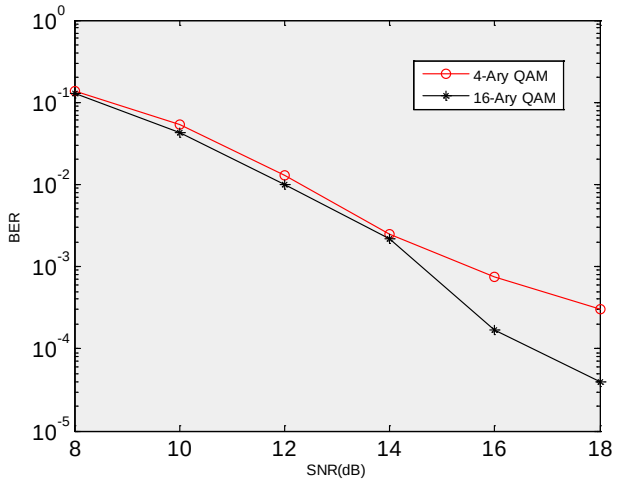


Fig.3 Flowchart of multiplicative neural network.

In the initial step of multiplicative filtering method of channel equalization of multiplicative neural network. In this process the input signal consist of the sum of desired signal , the channel non linearity and the interfering noise. A predefined delayed version of the original forms the training sequence to provide reference points for the adaptation process.

5. SIMULATION/EXPERIMENTAL RESULTS

To study the BER performances the equalizer structure was trained with 3000 iterations and tested over 10000 samples.



The data set has been pre-processed by normalizing them between 0.1 and 1. In all simulations, the results reported are the average of several runs in each case. The bit error rate for 4-Ary QAM and 16-Ary QAM has been compared for various SNR. From the below graph it is clearly seen that MNN provides better performance for 16-Ary QAM.

6. CONCLUSION

A high order feed forward neural network equalizer with multiplicative neuron is proposed in this paper. Use of multiplication allows direct computing of polynomial inputs and approximation with fewer nodes. Performance comparison BER performance suggest the better classification capability of the proposed MNN equalizer for 16-Ary QAM signals as compared to that of 4-Ary QAM

REFERENCES

[1] Nonlinear Stationary Channel Equalization of QAM Signals using Multiplicative Neuron Model IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.5, May 2011

[2] D.C. Park, M. A. El-Sharkawi, and R. J. Marks II, “Adaptively trained neural network,” IEEE Trans. Neural Networks, vol. 2, pp. 334–345, May 1991.

[3] Pham DT, Liu X, Neural networks for identification, prediction and control, London: Springer, 1995.

[4] S. Bang, S. H. Sheu, and J. Bing, “Neural network for detection of signals in communication,” IEEE Trans. Circuits Syst. I, vol. 43(8), pp. 644-655, Aug. 1996

[5] S. Chen, G. Gibson, C. Cown, and P. Grant, “Adaptive equalization of finite nonlinear channels using multilayer perceptrons,” Signal Processing, vol. 20, pp. 107-119, June 1990

[6] G. Gibson, S. Siu, and C. Cowan, “Multilayer perceptron structures applied to adaptive equalizers for data communications,” in Proc. ICASSP, May 1989, Glasgow, U.K., pp. 1183-1186

[7] T. Kim, T. Adali,” Fully complex multi-layer perceptron network for nonlinear signal processing,” J. VLSI Signal Process., vol. 32, No. 1, pp. 29-43, 2002.

[8] Q. Zhang, “Adaptive equalization using the back propagation algorithm,” IEEE Trans. Circuits Syst., vol. 37, pp. 848–849, June 1990.

[9] C.L. Giles and T. Maxwell,” Learning, invariance, and generalization in high-order neural networks,” Applied Optics, vol. 26, no. 23, pp. 4972-4978, 1987.

[10] E.M. Iyoda, K. Hirota, and F. J. Von Zuben,” Sigma-Pi cascade extended hybrid neural networks,” Journal of Advanced Computational Intelligence, vol.6, no. 3, pp. 126-134, 2002.

[11] M Schmitt,” On the complexity of computing and learning with multiplicative neurons,” Neural Comput. vol. 14, no.2, pp. 241-301, Feb. 2002.

AUTHOR'S PROFILE

Shivendra Kumar Tiwari has received his Bachelor of Engineering degree in Electronics & Communication Engineering from ASCT Engineering College, Bhopal in the year 2011. At present he is pursuing M.Tech. With the specialization of Digital Communication in IFIT Engineering College. His area of interest Communications, Electronics, Renewable energy sources, Signal and Systems.