A Compact Pi Network for Reducing Bit Error Rate in Dispersive FIR Channel Noise Model

Kavita Burse, R.N. Yadav, S.C. Shrivastava, Vishnu Pratap Singh Kirar

Abstract**—**During signal transmission, the combined effect of the transmitter filter, the transmission medium, and additive white Gaussian noise (AWGN) are included in the channel which distort and add noise to the signal. This causes the well defined signal constellation to spread causing errors in bit detection. A compact pi neural network with minimum number of nodes is proposed. The replacement of summation at each node by multiplication results in more powerful mapping. The resultant pi network is tested on six different channels.

Keywords**—**Additive white Gaussian noise, digital communication system, multiplicative neuron, Pi neural network.

I. Introduction

THE most basic elements of a communication system is a source of signal, a channel which introduces distortion as well as noise and a receiver with a means of detecting errors caused by noise. As higher-level modulation becomes more desirable to cope with the need for high-speed data transmission, nonlinear distortion becomes a major factor, which limits the performance of communication systems. Additive Gaussian noise can disturb the digitally modulated signal during transmission. Additive superimposed noise normally has a constant power density and a Gaussian amplitude distribution throughout the bandwidth of a channel. The source in the communication system model generates a 4-QAM complex valued symbol set. The combined effect of the transmitter filter, the transmission medium, and additive white Gaussian noise (AWGN) are included in the channel. A widely used channel model is a finite impulse response (FIR) model whose output at time instant k is given by [1]:

$$a(k) = \sum_{i=0}^{n_{h-1}} h(i).t(k-i) \tag{1}$$

Where, $h(i)$ are the channel tap values and n_h is the length of the FIR channel. Artificial neural network (ANN) is well known for its ability of performing classification tasks. It is used at the receiver end for adaptive filtering of the signal and reducing the BER.

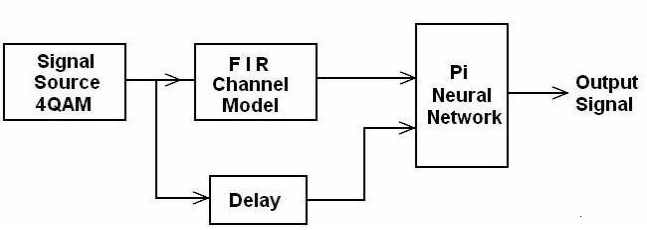

Fig. 1 A digital communication system model

This paper is organised as follows: Section II describes the pattern classification properties of ANN, in section III the learning rule for the pi neuron network is derived. Section IV discusses the experiments and the results and section V concludes the paper.

II. Pattern Classification Property of ANN

The ANN consists of many nodes called neurons with weighted interconnections (links) between them. The incoming signals (x_i) are multiplied by the corresponding weights (w_i) of the links and a bias term (b_i) is added. These terms are then multiplied to form the net input at the neuron, which is subjected to a nonlinear function like sigmoidal. The multiplicative neuron (π neuron) shown in figure 2 is defined as [2, 3]:

$$y = \prod_{i=1}^{n} (w_i x_i + b_i) \tag{2}$$

This differs by the standard perceptron defined by

$$y = \sum_{i=0}^{n} (w_i x_i + b_i) \tag{3}$$

Feedforward (FF) neural networks incorporating product terms are known to have more powerful mapping abilities. These multiplicative networks present better approximation capability and faster learning time as compared to multi layer perceptron (which employs additive neurons only) because of its processing of higher order information [4]. Multiplicative neuron models are mainly employed in higher order neural

Kavita Burse is a research scholar in Department of Electronics and Communication at Maulana Azad National Institute of Technology, Bhopal, India. (e-mail: profkavitaburse@ rediffmail.com).

Dr. R.N. Yadav and Dr. S.C. Shrivastava are with the Department of Electronics and Communication, Maulana Azad National Institute of Technology, Bhopal, India.(e-mail: yadavrn@manit.ac.in).

Vishnu Pratap Singh Kirar is with the Department of Electronics and Communication, Truba Institute of Engineering and Information Technology, Bhopal, India.(e-mail:vishnupskirar@live.com).

networks [5]. The network algorithm tunes the weights automatically to minimise an error function. This is implemented by learning from the past set of data fed to the network and to be able to apply this knowledge for future decision making (known as generalisation). The 4-QAM signal constellation is basically a 4 category classification problem, where the in phase and the quadrature component take on the values {-1, 1} + j*{-1, 1}, where j = sqrt (-1) [6]. The estimate of the transmitted signal with the help of a simple neuron can be computed as:

$$\hat{s}(t-d) = \hat{s}_R(t-d) + j\,\hat{s}_I(t-d) \tag{4}$$

$$\hat{s}(t-d) = \mathrm{Re}[F(y(t))] \tag{5}$$

$$\hat{s}(t-d) = \mathrm{Im}[F(y(t))] \tag{6}$$

d is the decision delay and $\hat{s}(t-d)$ is the estimate of the transmitted signal and F is a nonlinear function.

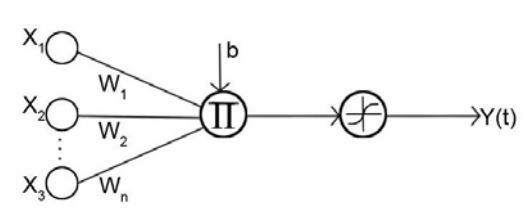

Fig. 2 A singe multiplicative neuron

III. Learning Rule for the Pi Neuron Network

The Pi neuron based ANN is shown in figure 3. In this network, each neuron first adds each of the weighted inputs in the space to an additional weight known as bias and then these summations are multiplied to provide a polynomial as the output of the neuron. This is fed to a bipolar sigmoidal activation function to create the final output. 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.

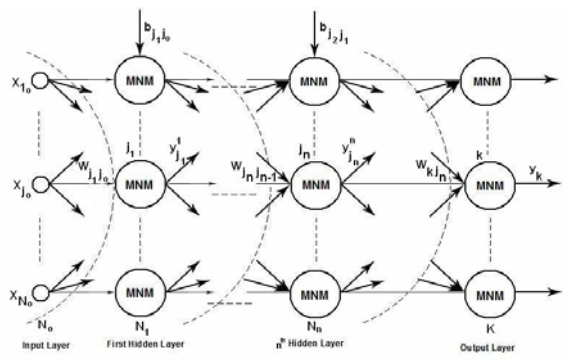

Fig. 3 Pi neuron based ANN

The symbols used are as follows:

N_o is the number of inputs in the input layer.

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

N_n 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}^{n} 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

$$y_{j1}^{1} = f\left(\prod_{j0=1}^{N0}(w_{j1j0}x_{j0} + b_{j1j0})\right) \tag{7}$$

for j_1=1,2,….,N_1 and x_{j0} represents j^{th} input in the input layer and f(.) is the activation function defined by $f(y) = \frac{1-e^{-y}}{1+e^{-y}}$

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

$$y_{j2}^{2} = f\left(\prod_{j1=1}^{N1} w_{j2j1}y_{j1}^{1} + b_{j2j1})\right); \text{ for } j_2=1,2,\ldots,N_2 \tag{8}$$

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

$$y_{jn}^{n} = f\left(\prod_{jn-1=1}^{Nn-1}(w_{jnjn-1}y_{jn-1}^{n-1} + b_{jnjn-1})\right); \text{ for } j_n=1,2,\ldots,N_n \tag{9}$$

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

$$E_{MSE} = \frac{1}{2PK}\sum_{k=1}^{K}\sum_{p=1}^{P}(y_{dk}^{p} - y_k^{p})^2 \tag{10}$$

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:

$$\Delta w_{kjn} = -\eta \frac{\partial E_{MSE}}{\partial w_{kjn}} = \eta \delta_k \frac{\left[\prod_{jn=1}^{Nn}(w_{kjn} y_{jn}^n + b_{kjn})\right]}{(w_{kjn} y_{jn}^n + b_{kjn})} \cdot y_{jn}^n \quad (11)$$

$$\delta_k = \frac{1}{PK}\left[\sum_{k=1}^{K}\sum_{p=1}^{P}(y_{dk}^p - y_k^p).\left[(1/2)(1+y_k^p)(1-y_k^p)\right]\right] \quad (12)$$

$$\Delta b_{kjn} = \eta \delta_k \frac{\left[\prod_{jn=1}^{Nn}(w_{kjn} y_{jn}^n + b_{kjn})\right]}{(w_{kjn} y_{jn}^n + b_{kjn})} = \frac{\Delta w_{kjn}}{y_{jn}^n} \quad (13)$$

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

$$\Delta w_{jnjn-1} = -\eta \frac{\partial E_{MSE}}{\partial w_{jnjn-1}} = \frac{\eta}{PK}\left[\sum_{k=1}^{K}\sum_{p=1}^{P}(y_{dk}^p - y_k^p).\frac{\partial y_k^p}{\partial y_{jn}^n}\right].\frac{\partial y_{jn}^n}{\partial w_{jnjn-1}}$$

$$= \eta \delta_k \frac{\left[\prod_{jn=1}^{Nn}(w_{kjn} y_{jn}^n + b_{kjn})\right]}{(w_{kjn} y_{jn}^n + b_{kjn})} \cdot w_{kjn} \cdot \frac{\partial y_{jn}^n}{\partial w_{jnjn-1}} \quad (14)$$

$$\Delta b_{jnjn-1} = \frac{\Delta w_{jnjn-1}}{y_{jn-1}^{n-1}} \quad (15)$$

Similarly, we can write equations for weight change between the hidden layer 1 and the input layer.

The weights and biases are updated as

$$w_i^{new} = w_i^{old} + \Delta w_i \quad (16)$$

$$b_i^{new} = b_i^{old} + \Delta b_i \quad (17)$$

IV. Experiments and Result

A. BER Plots

Six different channels were studied with the following normalised transfer function given in z-transform form:

CH1: 1.0

CH2: $0.447 + 0.894\,z^{-1}$

CH3: $0.209 + 0.995\,z^{-1} + 0.209\,z^{-2}$

CH4: $0.260 + 0.930\,z^{-1} + 0.260\,z^{-2}$

CH5: $0.304 + 0.903\,z^{-1} + 0.304\,z^{-2}$

CH6: $0.341 + 0.876\,z^{-1} + 0.341\,z^{-2}$

The channel CH1 has unity impulse response and no inter-symbol interference (ISI). CH2 corresponds to non minimum phase channel. CH3, CH4, CH5 and CH6 correspond to eigen value ratio of the input correlation matrix as 2.9, 3.1, 3.3 and 3.5 respectively. 3000 signal samples were used for training and 10000 signal samples were used for testing. The pi network consist of an input layer with 3 inputs, hidden layer 1 with 4 nodes, hidden layer 2 with 2 nodes and an output layer with 1 output. The corresponding number of node computations for MLP or Chebyshev network is much higher [7]. However as the nonlinearity in the input space increases the complexity of the pi network also increases. The BER for various channels at varying signal to noise ratio (SNR) is plotted in figure 4.

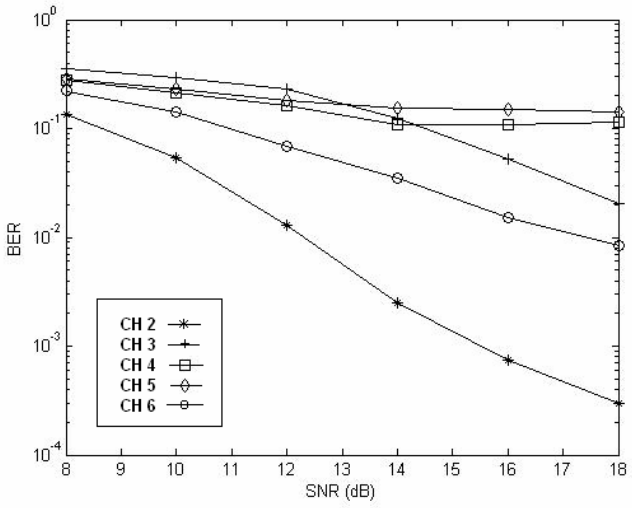

Fig. 4 BER for various channels

B. The eye patterns

The eye patterns of the output values provide an indication of effectiveness of the noise reduction. The eye pattern for CH2 at 12 dB SNR is shown in figure 5.

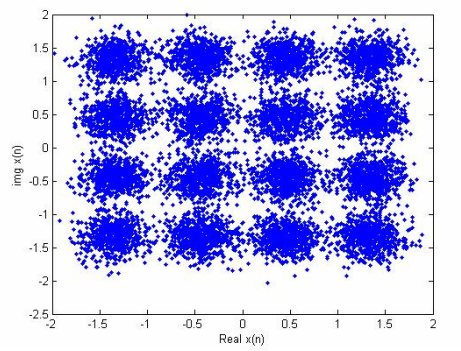

Fig. 5 (a) Nonlinear noisy channel output of 4 QAM signal

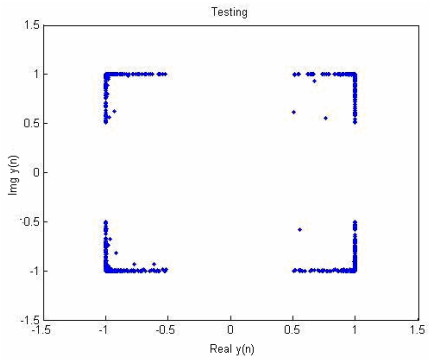

Fig. 5 (b) 4 QAM signal output of pi network

V. CONCLUSION

A compact pi network has been proposed for reducing BER in QAM signals transmitting through dispersive FIR channels. The simulation results for 6 different channels are plotted. It is observed that the multiplicative ANN structure is simple and fast for filtering noise even at very low values of SNR.

REFERENCES

[1] J.C. Patra, R.N. Pal, R. Baliarsingh and G. Panda, "Nonlinear channel equalization for QAM signal constellation using artificial neural networks," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 29 (2), pp. 262–271,1999.

[2] R. N. Yadav, V. Singh and P. K. Kalra," Classification using single neuron," in *Proc. IEEE Int. Conf. on Industrial Informatics*, Banff, Alberta, Canada, Aug. 21–24, 2003, pp. 124–129.

[3] R. N. Yadav, P. K. Kalra and J. John," Time series prediction with single multiplicative neuron model," *Applied soft computing*, vol 7, pp 1157-1163,2007.

[4] C.L Giles and T. Maxwell," Learning, invariance and generalization in high-order neural networks," *Applied Optics*, vol. 26(23), pp. 4972–4978, 1987.

[5] M. Schmitt," On the complexity of computing and learning with multiplicative neurons," *Neural Computing*, vol. 14(2), pp. 241–301, 2002.

[6] Kavita Burse, R.N. Yadav and S.C. Shrivastava," Complex Channel Equalization using Polynomial Neuron Model," in *Proc. IEEE 3rd Int. Symposium on Information Technology*, Kuala Lumpur, Malaysia, Aug. 26-29, 2008, pp. 771-775.

[7] Wan-De Weng, Che-Shih Yang, Rui-Chang Lin," A channel equalizer using reduced decision feedback Chebyshev functional link artificial neural networks," *Information Sciences*, vol. 177(13), pp. 2642-2654, 2007.