The Comparative Study of Adaptive Channel Equalizer Based on Feed Forward Back Propagation, Radial Basis Function Neural Network (RBFNNs) & Least Mean Square (LMS) Algorithm

Darshna Kundu (M. Tech Scholar)* M.K. Jain (Associate Professor)**

Abstract

Artificial Neural networks (ANNs) have been extensively used in many signal processing applications. Linear & Nonlinear adaptive filters based on a variety of neural network models have been used successfully for system identification in a wide range of applications. Due to their capacity to form complex decision regions, ANNs have been most popularly applied, in particular, for channel equalization of digital communication channels. The mean square error (MSE) criterion, which is usually adopted in neural learning, is of interest in the channel equalization. In this paper, we introduce a novel approach to adaptive channel equalization using feed forward & Radial Basis Function neural network (RBFNNs) that exploits the principle of discriminative learning while minimizing error function. The performance of proposed method has been compared with gradient based algorithms such as LMS (Least Mean Square) which are often characterized by slow convergence. ANNs technique using fast learning feed forward configuration based on Back Propagation algorithm, offers high speed of convergence w.r.t. LMS adaptive filtering algorithm. Computer simulation for the equalization of QAM signals in AWGN transmission channel is presented, which demonstrates the effectiveness of the proposed technique vis-à-vis LMS algorithm.

Keywords - LMS, MSE, ANN, RBF, Artificial Neural Network

I. INTRODUCTION

Artificial neural networks (ANNs) takes their name from the network of nerve cells in the brain. Recently, ANNs has been found to be an important technique for classification and optimization problem. Neural networks (NNs) have been extensively used in many signal processing applications. Linear & Nonlinear adaptive filters based on a variety of neural network models have been used successfully for system identification in a wide class of application. NNs have been most popularly applied to channel equalization of digital communication channels, in particular, due to their capacity to form complex decision regions. The mean square error (MSE) criterion, which is usually adopted in neural learning is of interest in the channel equalization.

ANNs are non-linear data driven & follows self adaptive approach as opposed to traditional model based methods. They are powerful tools for modelling, specially when the underlying data relationship is unknown. ANNs can identify and learn correlated patterns between input data sets and corresponding target values. ANNs can be used to predict...
the outcome of new independent input data after training

ANNs imitate the learning process of human brain & can
process problems involving non-linear & complex data even
if the data are imprecise & noisy.

A very important feature of these networks is their adaptive
nature, where “learning by example” replaces “programming”
in solving problems this feature makes such computational
model very appealing in application domain where one has
little or incomplete understanding of the problem to be solved
but where training data readily available.  ANN is capable of
performing nonlinear mapping between the input and output
space due to its large parallel interconnection between
different layers and the nonlinear processing characteristics.
Thus they are ideally suited for the equalization of AWGN
communication channel which are noisy as well as non-linear.

The basic structure of an artificial neuron is presented in
Fig.2. The operation in a neuron involves the computation of
the weighted sum of the inputs and threshold. The resultant
signal is then passed through a nonlinear activation
function. This is also called a perceptron, which is built around
a non-linear neuron.

The output of the neuron may be represented as,

\[ y(n) = \varphi \left( \sum_{j=1}^{N} w_j(n)x_j(n) + \alpha(n) \right) \]  

Where \( \alpha(n) \) is the threshold/bias to the neurons at the first
layer, \( w_j(n) \) is the weight associated with the \( j \)-th inputs to the
neuron and \( \varphi(.) \) is the nonlinear activation function. Different
kinds of nonlinear activation functions are shown in fig(5).

1.1 Neural Network layers:

The commonest type of ANN consists of three groups, or
layers, of units: a layer of "input" units is connected to a layer
of "hidden" units, which is connected to a layer of "output"
units(Fig.3). The activity of the input units represents the raw
information that is fed into the network. The activity of each
hidden unit is determined by the activities of the input units
and the weights on the connections between the input and the
hidden units. The behavior of the output units depends on the
activity of the hidden units and the weights between the
hidden and output units.

1.2 Neural Network Types:

1.2.1. Feed-forward Network:- Feed-forward ANNs (Fig.3)
allow signals to travel one way only; from input to output.
There is no feedback (loops) i.e. the output of any layer does
not affect that same or preceding layer. Feed-forward ANNs
tend to be straight forward networks that associate inputs
with outputs. They are extensively used in pattern
recognition. This type of organisation is also referred to as
bottom-up or top-down.
1.2.2 Radial basis function Network:-

Radial Basis Function Networks (RBFN) consists of 3 layers; an input layer, a hidden layer & an output layer. The hidden units provide a set of functions that constitute an arbitrary basis for the input patterns. Hidden units are known as radial centers and represented by the vectors $c_1, c_2, c_3, \ldots, c_h$. Transformation from input space to hidden unit space is nonlinear whereas transformation from hidden unit space to output space is linear. Dimension of each center for a p input network is $[p*1]$.

Different types of radial basis functions could be used, but the most common is the Gaussian function:

$$\varphi(v) = \frac{1}{1 + e^{-v^2}} \quad \text{(3)[15]}$$

For sigmoid units, the output varies continuously but not linearly as the input changes. Sigmoid units bear a greater resemblance to real neurons than do linear or threshold units, but all three must be considered with rough approximations.

This function is s-shaped, is the most common form of the activation function used in artificial neural network. It is a function that exhibits a graceful balance between linear & nonlinear behavior. It is represented as:

$$\varphi(v) = \frac{1}{1 + e^{-v^2}} \quad \text{(3)[15]}$$

Where $v$ is the input to the sigmoid function and ‘$a$’ is the slope of the sigmoid function. For the steady convergence, a proper choice of ‘$a$’ is required.
1.4 Multilayer Neural Network (Multilayer Perceptron)

In the multilayer neural network or multilayer perceptron (MLP), the input signal propagates through the network in a forward direction, on a layer-by-layer basis. This network has been applied successfully to solve some difficult and diverse problems by training in a supervised manner with a highly popular algorithm known as the error back-propagation algorithm.

The scheme of MLP using four layers is shown in below fig. 6 represents the input to the network, \( f_i \) and \( f_k \) represent the output of the two hidden layers and \( y_l(n) \) represents the output of the final layer of the neural network. The connecting weights between the input to the first hidden layer, first to second hidden layer and the second hidden layer to the output layers are represented \( w_{ij} \), \( w_{jk} \) and \( w_{kl} \) by respectively.

If \( P_1 \) is the number of neurons in the first layer, each element of the output vector may be calculated as,

\[
f_j = \phi_j \sum_{i=1}^{N} \left[ w_{ij} f_i(n) + \alpha_i \right], j = 1, 2, 3, \ldots, P_2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(4)\]

where \( \alpha_i \) is the threshold to the neurons at the first layer, \( N \) is the number of inputs and \( \phi_j \) is the non-linear activation function. The time index \( n \) has been dropped to make the equations simpler. Let \( P_2 \) be the number of neurons in the second layer. The output of this layer is represented as, \( f_k \) and may be written as

\[
f_k = \phi_k \sum_{j=1}^{P_2} \left[ w_{jk} f_j(n) + \alpha_k \right], j = 1, 2, 3, \ldots, P_2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(5)\]

Where, \( \alpha_k \) is the threshold to the neurons in the second layer. The output of the final layer can be calculated as

\[
y_l(n) = \phi_l \sum_{k=1}^{P_3} \left[ w_{kl} f_k(n) + \alpha_l \right], j = 1, 2, 3, \ldots, P_3 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(6)\]

Where, \( \alpha_l \) is the threshold to the neuron at the final layer and \( P_3 \) is the number of neurons in the output layer. The output of the MLP may be expressed as

\[
y_l(n) = \phi_l \left[ \sum_{k=1}^{P_3} w_{kl} f_k(n) + \alpha_l \right] + \alpha_l \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(7)\]

II. METHODOLOGY

2.1 Adaptive Algorithms Used For Optimization

2.1.1 Back-propagation (BP) Algorithm

In BP algorithm, initially the weights and thresholds are initialized as very small random values. The intermediate and the final outputs of the MLP are calculated by using (4), (5) and (6).

The final output \( y_1(n) \) at the output of neuron 1, is compared with the desired output \( d(n) \) and the resulting error signal \( e_1(n) \) is obtained as

\[
e_1(n) = d(n) - y_1(n) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(8)\]

The instantaneous value of the total error energy is obtained by summing all error signals over all neurons in the output layer, that is

\[
\xi(n) = 1/2 \sum_{i=1}^{P_3} e_i^2(n) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(9)\]
where P3 is the number of neurons in the output layer

This error signal is used to update the weights and thresholds of the hidden layers as well as the output layer. For measuring degree of matching, MSE (mean square error) performance criteria is taken into consideration.

The updated weights are,

\[ w_{kl}(n+1) = w_{kl}(n) + \Delta w_{kl}(n) \] ……..(10)[15]

\[ w_{jk}(n+1) = w_{jk}(n) + \Delta w_{jk}(n) \] ……..(11)[15]

\[ w_{ij}(n+1) = w_{ij}(n) + \Delta w_{ij}(n) \] ……..(12)[15]

Where, \( \Delta w_{kl}(n) \), \( \Delta w_{jk}(n) \), and \( \Delta w_{ij}(n) \), are the change in weights of the output, hidden and input layer respectively.

The thresholds of each layer can be updated in a similar manner, that is

\[ a_i(n+1) = a_i(n) + \Delta a_i(n) \] ……..(13)[15]

\[ a_k(n+1) = a_k(n) + \Delta a_k(n) \] ……..(14)[15]

\[ a_j(n+1) = a_j(n) + \Delta a_j(n) \] ……..(15)[15]

**Steps of the BP Algorithm:**

Step 1: obtain a set of training data

Step 2: set up neural network model

[Number of input neurons, Hidden neurons, and Output neurons, number of layers]

Step 3: Set learning rate \( \eta \) and momentum rate \( \alpha \)

Step 4: initialize all connection weights \( w_{kl}, w_{jk}, w_{ij} \) and bias values \( a_i, a_k, a_l \)

Step 5: set minimum error, \( E_{min} \)

Step 6: start training by applying input data and propagate through the layers then calculate total error.

Step 7: Back propagate error through output & hidden layer and adapt weights.

Step 8: Back propagate error through input & hidden layer and adapt weights.

Step 9: check it error is \( < E_{min} \)

If not …. repeat step 6-9, if yes….stop training.

**2.2.2 LMS (standard LMS) algorithm:**

The least-mean square (LMS) algorithm updates the linear filter coefficients such that the mean square error (MSE) cost function is minimized.

**Fig: (9) Basic Block diagram of Adaptive Channel Equalizer Based on gradient decent algorithms [17]**

LMS perform the following operation to update coefficients of the adaptive filter.

\( u(n) \) = Input signal

\( h(n) \) = system response

\( e(n) \) = Error signal

\( d(n) \) = desired signal.

\( \hat{a}(n) \) = Transversal FIR filter output

\( \wp(n) \) = weight vector of Transversal FIR filter

\( \xi(n) = E[e(n)] \) …………………(16)[4]

The error estimation \( e(n) \) is

\( e(n) = d(n) - \hat{a}(n) \) …………………(17)[4]

\( \hat{a}(n) = w(n) * u(n) \) …………………(18)[4]

Calculates the error signal \( e(n) \) by using the equation (2).

Coefficient updating equation is

\( w(n+1) = w(n) + \mu u(n) e(n) \) ………(19)[4]
Where $\mu$ is the step size of the adaptive filter, $n$ is the weight vector, and $u(n)$ is the input signal vector.

### III. IMPLEMENTATION

3.1 Designing ANN models

Designing ANN models follows a number of systemic procedures. In general, there are five basics steps: (1) collecting data, (2) preprocessing data, (3) building the network, (4) train, and (5) test performance of model as shown in fig(10).

![Fig. 10](image_url)

3.2 Set up for performance measurement: LMS, ANN based equalizer
IV. SIMULATION RESULTS

During simulation mean squared error (MSE) was used as performance criteria. This section presents the MSE performance of equalizer based on LMS for variety of parameters and neural network equalizer using back propagation algorithm to make a comparative analysis in terms of MSE performance criteria. The MSE versus Number of Iterations for AWGN Channel with Eb/No=1dB using QAM as modulation technique at receiver i/p was plotted for the performance analysis.

Simulation 1: Computer simulation of LMS based equalizer

Simulation 2: Computer simulation of feed forward two layer neural network based Equalizer trained using Levenberg-Marquardt Back propagation algorithm (trainlm) by varying number of parameters like input & target data size, number of hidden neurons while selecting AWGN channel with (Eb/No=1dB) & maintaining percentage of validation & testing data as 15%.
In the above section, computer simulations are carried out & plots are drawn between MSE & number of iterations for performance analysis. Here each learning curve is the result of ensemble averaging the instantaneous squared error “e(t)” [MSE] versus “number of iterations”.

In simulation of LMS based equalizer(fig.12) the results confirm that the rate of convergence of the adaptive equalizer based on LMS is highly dependent on the step-size parameter $\mu$. For a large step-size parameter ($\mu=0.01$), the equalizer converged to steady-state conditions in approximately 250 iterations(convergence faster). On the other hand, when $\mu$ was small (=0.001), the rate of convergence slowed down to 500 iterations. The result also shows that the mean square error
(MSE) has lower value of $10^{-2}$ at $\mu=0.01$(larger step size) than MSE of $10^{-0.5}$ at $\mu=0.001$(smaller step size).

Simulation results (Fig.13) relating to equalizer using feed forward two layer neural network presents a plot of MSE versus number of epochs(number of iterations), for data size [10 10], with different number of hidden neurons i.e.10,20,50&100.

In the plot of 10 hidden neurons by training a neural network, we got MSE of $10^{-12}$ within 5 epochs, in a plot of 20 neurons we got MSE value $10^{-15}$ in 4 epochs, & in a simulation of model with 50 hidden neurons MSE value achieved is $10^{-20}$ by taking 4 epochs only, and with 100 hidden neurons we got MSE $10^{-16}$(approx.) in 3 epochs. From the study of simulations, it can be noticed that with the increase of number of hidden neurons (doubled) there is no very far reduction in MSE value, but computational complexity alongwith computation time increases upto a large extent..

Fig.14 depicts performance in terms of MSE versus number of iterations (epochs) of the equalizer based on ANN using RBF. It can be seen that we got MSE $10^{-20}$(approx.) in 3 epochs only.

**VI. CONCLUSION**

Adaptive equalizer based on Artificial neural network employing network configuration such as Radial Basis function & feed forward using back propagation algorithm for different values of parameters such as data size, number of hidden neurons are simulated. In parallel to this gradient based LMS equalizer is also simulated & plots are obtained for different values of step size & decimation factor. On comparative analysis, we infer that a neural network model, as simulated above, yields a far reduction in mean squared error MSE with reduced convergence time compared to LMS based equalizer.

**VII. REFERENCES**


[9]. O.Macchi “Adaptive processing, the least mean squares approach with applications in transmission” West Sussex: John Wiley and Sons, 1995


[15]. Debidutta Mohanty “Channel equalization using GA family”National Institute of technology Rourkela(orissa),2008

[16]. “Recursive Least Squares Adaptive Filters a better ISI Compensator” Sharma, O., Janyani, V and Sancheti, S. 2009 International Journal of Electronics, Circuits and Systems

[18]. M.S.Chavan, R.H.Chile “Optimizing the Performance of Fuzzy Implemented Radial Basis Function Channel Equalizer with Subset Centre Selection" *International journal of emerging trends technology in computer science* Volume 1 Issue 1, May-June 2012 ISSN 2278-6856

[19]. Maitha H. Al Shamisi, Ali H. Assi and Hassan A. N. Hejase “Using MATLAB to Develop Artificial Neural Network Models for Predicting Global Solar Radiation in Al Ain” *City – UAE United Arab Emirates University United Arab Emirates*

*Darshna Kundu is currently pursuing M.Tech from Lingayas University, Faridabad, Haryana, India.*

Email: darshanakundu@gmail.com

**M.K.Jain is an Associate Professor currently working with Lingayas University, Faridabad, Haryana, India.*

Email: mkjain@lingayasuniversity.edu.in