International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 18 (2018) pp. 13409-13420

Investigating Signal Power Loss Prediction in A Metropolitan Island Using ADALINE and Multi-Layer Perceptron Back Propagation Networks

Virginia Chika Ebhota[1*], Joseph Isabona[2], Viranjay M. Srivastava[3]

[1, 2, 3]Department of Electronic Engineering, Howard College, University of KwaZulu-Natal, Durban, South Africa. Correspondence author

Abstract

The prediction ability of every artificial neural network model depends on number of factors such as the architectural complexity of the network, the network parameters, the underlying physical problem, etc. The performance of two ANN architectures: ADALINE and MLP network in predicting the signal power loss during electromagnetic signal propagation using measured data from LTE network was explored in this work. Learning rate parameter was used to the check the predictive effectiveness of the two network models while considering the prediction error between the actual measurement and the prediction values using four performance error indicators. The gradient and momentum parameters of the two networks were also checked at different variation of learning rate. ADALINE show the best prediction ability at 0.1% learning rate but has the best gradient of 0.074406 which approximates to the local minima at 0.9% learning rate. The reverse was the case for MLP as it best predicted the actual measurement at learning rate of 0.9% but the gradient was farther from the local minima at 0.9% learning rate. There is therefore need for appropriate choice of learning rate that will be high enough to increase convergence and but not too high to result to system overshoot. Training the two ANN network with different combination of linear and non-linear transfer functions shows effective performance of hyperbolic tangent and logistic sigmoid in the hidden and output layers of MLP and purelin transfer function performed best with ADALINE which is in agreement.

Keywords: Adaptive linear element, Multi-layer perceptron neural network, Back propagation, Learning rate, Gradient descent, Momentum parameter, Input delay, Transfer functions.

INTRODUCTION

Artificial neural networks (ANNs) are systems consisting of computationally simple but non-linear element which are parallel inter-connected and have the capability of processing information from the environment [1]. Different numbers of artificial neural network architectures have been developed over the decades. These include Multi-layer feedforward networks developed by Rumelhart et al in 1986, Self organizing feature network maps developed by Kohonen in 1982, Hopfield networks developed by Hopfield in 1987, Radial Basis Function network developed by Powell in 1987, Counter-propagation network developed by Hecht-Nielsen in 1987, and Recurrent ANNs developed by Elman network: Elman in 1988 [2-7]. The network architecture can be totally or sparsely connected, binary or floating point in operation, asynchronous or synchronous, static or dynamic, discrete or continuous and adopt supervise or unsupervise learning. However, the effort to make choice among the various ANN architectures and establish which is superior likely fails in most cases, as the choice should be application oriented [1]. It is preferred to test various types of ANNs for every novel application than making use of preselected one that performed well for a particular problem. Prior to the training of ANNs, it is important that the network architecture is well specified as the learning ability of the network and its performance is dependent on how suitable the architecture is for the case under study. If the ANN architecture is small, it may not have sufficient degree of freedom to capture fully the underlying relationships in the data while too large ANN architecture may lead to poor generalization. To overcome this problem, effective methods that will offer an unbiased ANN architecture for the underlying case of study is required should be use utilize.

Therefore, ANNs with parameters just enough to offer adequate fit with the data for the case under study are preferred to avoid over- fitting. In this work, two different ANN architectures, ADALINE and Multi-layer perceptron (MLP) are studied with measured data from a Long-Term Evolution (LTE) network to measure their performances in the prediction of signal power loss from the chosen area. The work examined the effect of input delay in ADALINE in comparison to MLP network using their learning rate as a benchmark. Also, the gradient of these models was captured at different learning rate.

ADAPTIVE LINEAR NEURON OR ELEMENT (ADALINE)

ADALINE is a linear single layer ANN that is based on McCulloch Pitts neurons and makes use of a linear activation function. It is made up of a weight, bias and a summation function with multiple nodes and each of the nodes takes multiple inputs and gives an output [8]. If x is an input vector to the neural network and n is the number of inputs, w and θ are the weight vector and a constant respectively and y is the model output, then, the output of ADALINE neural network is given as:

$$y = \sum_{i=1}^{n} x_i w_i + \theta \quad (1)$$

If $x_0 = 1$, and $w_0 = \theta$, the output is reduced to:

$$y = \sum_{i=0}^{n} x_i w_i \tag{2}$$

Assuming that the learning rate η is a positive constant, y and o_T is the output of the model and the target output respectively, the weight is then updated as:

$$w \leftarrow w + \eta\left(o_T - y\right)x \tag{3}$$

The ADALINE converges at $E = \left(0_T - y\right)^2$ which is the least square error.

The supervised learning of ADALINE is analogous to perceptron learning algorithm that is in use in standard McCulloh-Pitts perceptron such as Single layer perceptron network and Multi-layer perceptron network. However, they have similarities but there are significant differences as shown in table 1. Their similarities include:

i. They both have linear decision boundary.
ii. They can both learn iteratively, while the Perceptron networks learns naturally, ADALINE networks learn by means of stochastic gradient descent.
iii. Both are used for binary classification

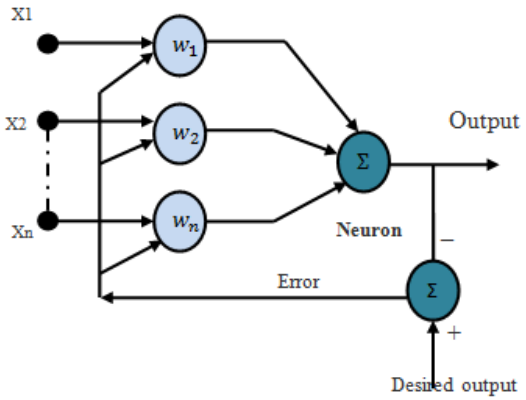

Figure 1. Architecture of an ADALINE

Training mechanism of ADALINE

The ADALINE structure is equivalent to a linear neuron that has an additional feedback loop and is determined by normalized least mean square training law (LMS) shown in figure 1. The input vectors i.e. the desired output is given to the network during the training phase and the weight adjusted adaptively based on least square rule. Once the weights are adequately adjusted, the activation function is then utilized at the training phase. The response from the training unit can be tested by the application of varieties of inputs that are not in the training set. When the network gives high degree of consistent response with the test inputs, the network could be said to generalize well. Training process and generalization are two major features of ADLINE network.

MULTI-LAYER PERCEPTRON (MLP) NETWORKS

Multi- layer Perceptron (MLP) networks are simply made up of networks of multiple neurons on multiple layers. The activation function here is not linear like that of ADALINE but MLP network makes use of non- linear activation function as logistic sigmoid, the hyperbolic tangent and the rectifier linear unit [9]. A perceptron is the simplest possible neural network with a computational model of a single neuron invented by Frank Rosenblatt in 1957. It is made up of one many inputs, a processor and an output. It is a unit that has weighted inputs which produces binary output based on threshold. Perceptron networks are trained by simple learning algorithms known as least square method or delta rule. This calculates error between the output of the sample data and the network output and makes use of it in creating an adjustment to the weights thereby implementing a sort of gradient descent. The most fundamental type of an activation function is a binary function with only two possible results.

$$f(x) = \{1 if w * x + b > 0 \tag{4}$$

Otherwise, w is a vector of the weights which is real value, $w * x$ is the point product $\sum_{i=1}^{n} x_i w_i$, n is the inputs number and b is the bias. The function from equation (4) returns 1 if the input is 0 or positive and returns 0 for an input that is negative.

A neuron with the above type of activation function is known as Perceptron. The perceptron network is also known as the single layer Perceptron network and the simplest of the fedforward neural network. It can learn only linearly separable patterns as shown by Marvin Minsky and Seumour Papert in 1969. The Perceptron algorithm is summarized as follows:

i. For every input, it is multiplied by its weight
ii. Summation of all the weighted inputs
iii. The output of the perceptron is computed based on the sum that is passed through an activation function.

However, single layer Perceptron network are not capable of solving problems that are not linearly separable. Also, because a Perceptron network is linear, it will not be able to reach a state with all the input vectors appropriately trained with non-linearly separable training set i.e. if a hyper plane cannot be used to separate positive examples from negative examples. This will lead to non-convergence of the perceptron network. By connection of neurons in multi-layers in the network with non-linear activation function, non-linear decision boundaries which permit the solving of problems that are more complicated and non-linearly separable are created.

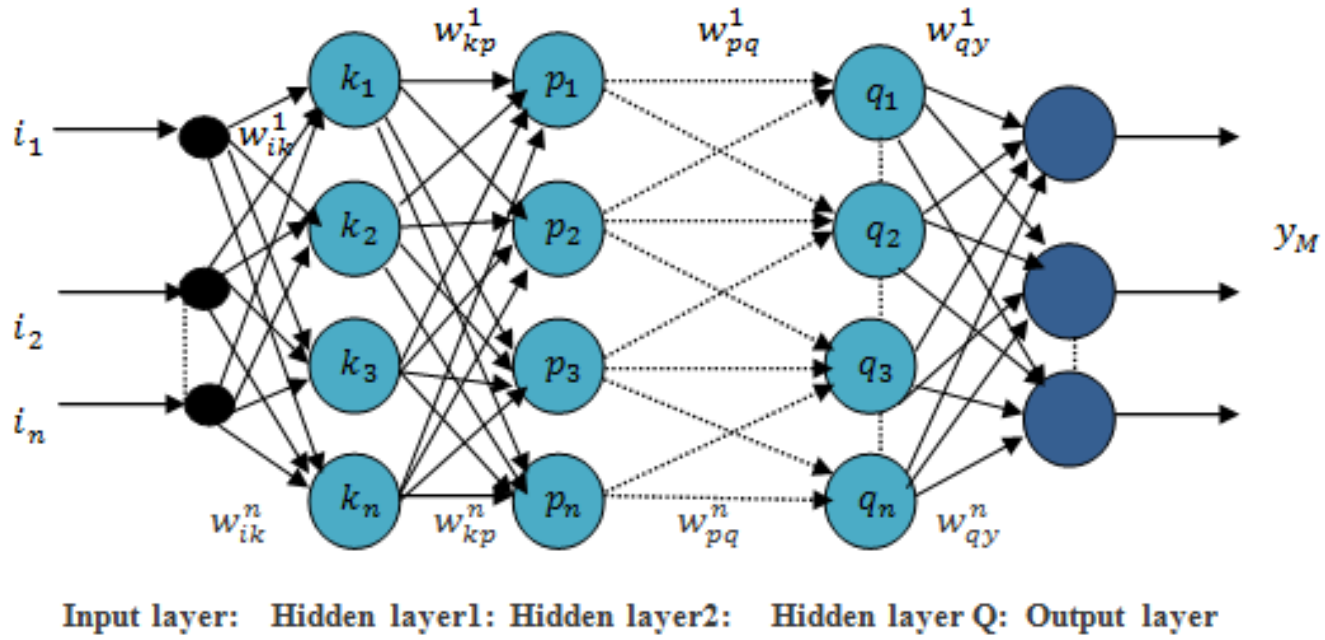

Figure 2: Architecture of a Multi-Layer Neural Network with n-hidden layer

Assuming input layer with i_n neurons, $i=(i_0,i_1,i_2,....i_n)$ as shown in figure 2, the network output at the first hidden layer k is computed as:

$$k_1^n = f\left(\sum_{k=1}^{n} w_{ik} i_1^n\right) \tag{5}$$

$n=1,2,3.....$ and $w_{ik}{}^n$ = the weight between the i and k neurons

This becomes the input to the second hidden layer p and the network output at the second hidden layer becomes:

$$p_1^n = f\left(\sum_{p=1}^{n} w_{kp}^n k_1^n\right) \tag{6}$$

This becomes an input to the Q hidden layer and the output of the Q hidden layer becomes:

$$q_1^n = f\left(\sum_{q=1}^{n} w_{pq}^n p_1^n\right) \tag{7}$$

Assuming a sigmoid activation function is used, thus:

$$f(x) = \frac{1}{1+e^{-x}} \tag{8}$$

The network overall output y_M is computed as:

$$y_M = f\left(\sum_{y=1}^{n} w_{qy}^n q_1^n\right) \tag{9}$$

Where $y_M = y_1 + y_2..... \; y_n = f(w,i)$

Where f = the transfer function and w = the matrix weight defined as: $w=(w_0,...w_{ik}{}^n,....w_{kp}{}^n,....w_{pq}{}^n,....w_{qy}{}^n) \in \Re$

Using multiple layers of hidden neurons in the hidden layer facilitates better processing power and flexibility of the system. The extra flexibility is because of the additional network complexity due to the multiple hidden layers in the training algorithm. However, too many hidden neurons normally result to over specification of the system thereby making it incapable of generalization, while few hidden neurons result to improper fitting of the input data by the system thereby reducing system robustness.

Architectural definitions of MLP networks are very important as lack of adequate choice of layers and neurons for connection can prevent the network from solving problems of inadequate adjustable parameters while excess choice of connection layers and neurons may result to over-fitting of training data. Architectural optimization of the connection of the hidden layers and neurons for establishing an ANN that effectively solve a given problem remains one of the tasks yet to be solved in many research areas[10].

ADALINE and Perceptron Artificial Neural Network

Differences	ADALINE neural network	Multi-layer perceptron (MLP)
Learning algorithm	The learning algorithm is based on the adjustment of the neurons weight for each weighted summation of the network inputs. The neuron in the network takes more than a single input but generates one output.	Uses perceptron rule as the learning algorithm where the neuron weights and biases are trained in a way that a correct target is produced on entering of the inputs.
Checks during network training	After every iteration, it checks if the weight works for the all the input patterns.	Just weight update after every new input pattern, no further checks are carried out.
Accuracy	More accurate in network training since it gives information on the suitability of the of the neuron for a given set of input pattern as the weights are continuously updated	Does not give information on the suitability of the neuron as the neurons are trained and there is no continuous update.
Learning method	Does not use derivative of transfer function. Instead, it makes use of continuous values predicted from the network input to learn model coefficient. This notifies on the degree of correctness or incorrectness	Uses derivative of transfer functions in the computation of changes in weight or class labels in the learning of model coefficient.
Transfer function	Uses linear transfer function like purelin	Uses non-linear transfer function like logistic sigmoid or hyperbolic tangent
Network architecture	A single layer of artificial neurons	Network or multiple artificial neurons over multiple layer which creates complex non- linear decision boundaries that permits solving of problems that are not linearly separable

THE CONCEPT OF BACK PROPAGATION

Back propagation (BP) determines the gradient of the loss function and is essentially applied in gradient descent optimization algorithm for weight adjustment of every neuron that contributed to the process of learning [11, 12]. For every supervised learning algorithm, the aim is to discover function which maps appropriately set of inputs to the corresponding right output. The essence of back propagation is to train network to adequately learn internal representations to permit it to learn any random input to output mapping [13].

Before the network training commences, the weights are randomly selected, and the neurons learn from the training examples. If $x_{i1}.....x_{in}$, 0_t are the training examples with $x_{i1}.....x_{in}$ being the network inputs and 0_t being the correct output which is expected from the network given the inputs, then $x_{i1}.....x_{in}$ will likely compute an output y_M different from the expected or desired output 0_t considering random weights. To measure the discrepancy of the desired output 0_t and the actual output y_M, squared error measure is applied as:

$$E = \left(0_t - y_M\right)^2 \tag{5}$$

Where E = error or discrepancy

The difficulty of inputs to outputs mapping can therefore be cut down to an optimization problem by locating a function which will give minimal error as shown in figure 4.

$$y_M = w_{i1},.....w_{in} \tag{6}$$

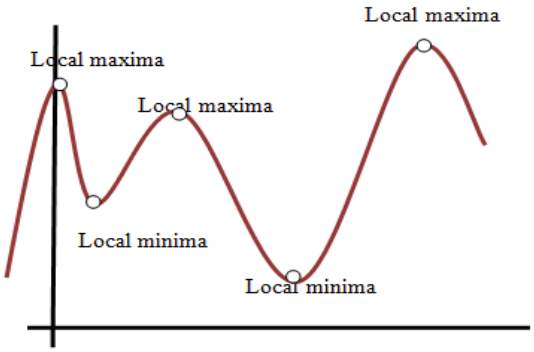

Figure 3. A sketch of a Function with local minima and Local maxima

The error is also dependent on the neuron weights which are eventually needed to be modified in the network to permit learning. A conventional algorithm used in finding the weights set which minimizes the error is the gradient descent algorithm while back propagation is applied in the calculation of the direction of the steepest descent [14].

Gradient Descent

Neuron weights permits ANNs to learn by updating after the forward passes of data through the network and the reason for weight adjustment is to ensure reconciliation of the difference between the actual and the predicted values ensuing forward passes. Error is an important measure to ascertain the differences and the respective error of each neuron send backward through the network to aid the process of update, i.e. back propagation of error.

Cost function for error determination based on neuron weights as shown in figure 5 below can be applied and the lowest point on the cost function known as the optimal value i.e. local minima where rate of function change equals zero can be ascertain. Conceptual use of slope of the angle of the cost function at the present location reveals the direction of the slope [15]. Algebraically, a negative slope indicates a downward movement while a positive slope indicates an overshoot i.e. movement beyond the optimal. The slope is determined using gradient descent. Gradient is the rate at which a function changes while descent implies exploring at the base of the cost function with the changing gradient. Gradient descent takes into account a total of data set forward pass and calculates cost and thereafter propagates the errors backward to the neurons through the network [16]. There are vanilla plain and stochastic determined gradient descents. All data weights are repeatedly adjusted using vanilla plain gradient descent while stochastic (SGD) gradient descent samples data randomly. Learning can be speed up by data random sampling for an improve prediction result. Gradient descent shows vulnerability to local minima if all data instance is applied in weight adjustment determination and may be made less vulnerable to extremes and outliers by considering the data en bloc, however this is undesirable when in search for global minima.

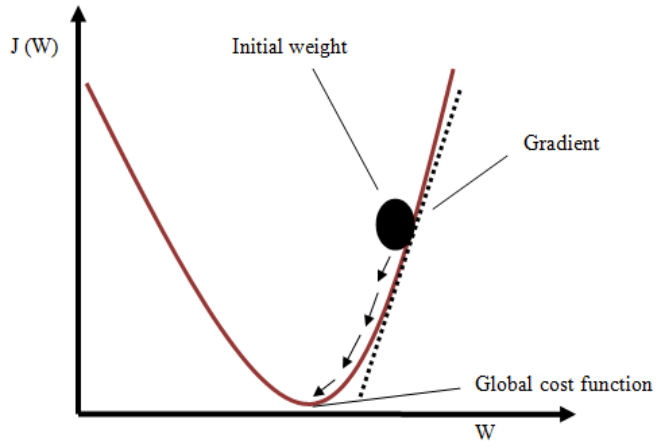

Figure 4. A sketch of Cost Function

The method of gradient descent comprises the calculation of the derivative of squared error function with respect to the network weights. This is achieved using back propagation. Squared error function is expressed in equation (7) considering an output neuron:

$$E = \frac{1}{2}\left(0_t - y_M\right)^2 \tag{7}$$

Where E = the squared error, 0_t = the desired or expected output, y_M =the actual output, $\frac{1}{2}$ cancels the exponent during differentiating.

For each neuron i, the output 0_k is defined as:

$$0_k = \varphi\left(net_i\right) = \varphi\left(\sum_{k=1}^{n} w_{ik} 0_k\right) \tag{8}$$

The input net_i is the weighted sum 0_k of output of preceding neurons, if net_i is in the first hidden layer, then 0_k = inputs x_i to the network. n = the neuron input units number, w_{ik} = weights between i and k neurons, φ = activation function which is non-linear and differentiable such as logistic function [17].

$$\varphi\left(z\right) = \frac{1}{1+e^{-z}} \tag{9}$$

The calculation of partial derivative of the error E with respect to the weights between neurons w_{ik} is performed by the use of chain rule twice as shown below:

$$\frac{\partial E}{\partial w_{ik}} = \frac{\partial E}{\partial 0_k}\frac{\partial 0_k}{\partial net_i}\frac{\partial net_i}{\partial w_{ik}} \tag{10}$$

From equation 10, a single term in the sum net_i is dependent on w_{ik}, therefore:

$$\frac{\partial net_i}{\partial w_{ik}} = \frac{\partial}{\partial w_{ik}}\left(\sum_{k=1}^{n} w_{ik} 0_k\right) = \frac{\partial}{\partial w_{ik}} w_{ik} 0_i = 0_i \tag{11}$$

$0_i = x_i$ for first layer neurons.

Assuming the use of a logistic function, output derivative of the neuron k with respect to input is described as:

$$\frac{\partial 0_k}{\partial net_i} = \frac{\partial}{\partial net_i}\varphi\left(net_i\right)\left(1-\varphi\left(net_i\right)\right) \tag{12}$$

This is the purpose for which back propagation requires an activation function to be differentiable. For output layer neuron, then the evaluation is straightforward as:

$o_k = y_M$ and

$$\frac{\partial E}{\partial 0_k} = \frac{\partial E}{\partial y_M}\frac{\partial}{\partial y_M} = \frac{1}{2}\left(0_t - y_M\right)^2 = y_M - 0_t$$

However, if k is inner layer of the network, it is less obvious finding the derivative E with respect to 0_k. In view of E being function of inputs of all neurons $N = p, q, ...r$ receiving input from the neuron k ,then:

$$\frac{\partial E\left(0_k\right)}{\partial 0_k} = \frac{\partial E\left(net_p, net_q, ...net_r\right)}{\partial 0_k} \tag{13}$$

Taking the sum derivative considering 0_k, a recursive example is gotten for the derivative as:

$$\frac{\partial E}{\partial 0_k} = \sum_{i\in N}\left(\frac{\partial E}{\partial net_i}\frac{\partial net_i}{\partial 0_k}\right) = \sum_{I\in N}\left(\frac{\partial E}{\partial 0_i}\frac{\partial 0_i}{\partial net_i}w_{ik}\right) \tag{14}$$

This can be a calculation of the derivative with respect to 0_k if every derivative with respect to the next layer output 0_i, the derivative nearer to the output neuron is known.

Their summation gives:

$$\frac{\partial E}{\partial w_{ik}} 0_i \delta_k \tag{15}$$

$$\delta_k = \frac{\partial E}{\partial 0_k}\frac{\partial 0_k}{\partial net_i}\begin{cases}\left(O_k - O_t\right)O_k\left(1-O_k\right) \ for\, k = output\, neuron \\ \sum_{t\in N}\delta_t w_{ik})O_k\left(1-O_k\right) \ for\, k = inner\, neuron\end{cases}$$

However, gradient descent algorithm with back propagation is not a guarantee to achieving the global minimum, it merely guarantees a local minimum and display problem of crossing plateau in the scenery of error function. These problems of non convergence of the error function might limits the performance of gradient descent with back propagation including its non requirement of input vector normalization as normalization improves network performance [18, 19].

Momentum Parameter

The momentum parameter is applied in preventing a neural network system from converging to a saddle point or local minimum. It simply adds a fraction of the preceding weight to the present weight. High momentum parameter helps to increase the convergence speed of the network system, however, momentum that is too high, creates the risk of system overshooting the minimum thereby causing instability of the system while very low momentum slows down the training of the network system and cannot dependable prevent local minima [20].

Applying momentum α which is a variable inertia term, the gradient descent and the last changes in weight can be weighted in a way that the adjustment of the weight in addition is dependent on the preceding change. A zero (0) momentum results from alteration from the gradient while a momentum of one (1), depends on the last weight change.

$$\Delta w_{ik}\left(t+1\right) = \left(1-\alpha\right)\eta\Delta w_{ik}\left(t\right) \tag{16}$$

Where, $w_{ik}\left(t+1\right)$ in the connection of k and i neurons at time $\left(t+1\right)$, $\Delta w_{ik}\left(t+1\right)$ = the change in weight, η = a learning rate, δ_k = error signal of neuron k. 0_i = neuron i output, α = the inertial term.

Momentum is dependent on present weight change $\left(t+1\right)$, together with the present gradient of error function and the changes in weight from the previous point in time. It resolves the problem of getting stuck in steep ravines and flat plateau by deceleration of gradient descent as it get small in flat plateau to ensure quicker escape.

Learning Rate Parameter

The learning rate parameter controls the weights and biases of network during the network training using the training algorithm by updating the weights of the neuron using the gradient descent [21]

$$\Delta w_{ik} = \eta\frac{\partial E}{\partial w_{ik}} = \eta O_k \delta_k \tag{17}$$

Multiplying by -1 ensures update in the minimum direction of the error function to examine oscillation such as alternation of weights and thereby improving the rate of convergence.

Delay Parameter

This parameter enhances the recognition of pattern place i.e. the time invariant by storing the older activation and the connection values of the feature elements. This is carried out by re-copying of the feature elements with every out going connection in every time step before the update of the original elements. The totality of the number of time steps saved by the procedure is known as delay. The effect of input delay in the ADALINE neural network was checked in this work against the MLP neural network without delay.

METHOD OF DATA COLLECTION

We carried out the research work in Bonny Island, an urban area in Rivers State, Nigeria. Measured data were collected via drive test.

Study Area

Bonny Island is an urban area and a local government area in Rivers State, Nigeria. It is a major oil export point as most of the extracted onshore oil in River State is piped to Bonny for its export. It has the biggest Nigeria liquefied natural gas

plant with six trains [22]. The area covers a distance of 645.60km^2 with a population of 214,983 from Nigeria national population census carried out in 2006. Bonny Island is situated at the edge of the Atlantic ocean and has coordinates of 4^o 26'N 7^o 10'E [23]. It comprises both territorial areas and virgin lands and the water taxis that moves along the bonny river provides the connection between Port Harcourt city and Bony Island [24]. An online area map of Bonny Island is shown in figure 5.

Figure 5: A view of Bonny Island (adapted from Weinstein, 2009)

Collection of Data

The collection of data was done by means of a drive test from a chosen base station transmitting at 1900MHz frequency. Information on the cell file was provided by the mobile network provider. During the drive test, a laptop and a sonny mobile handset installed with 2016 test mobile system (TEM) software, global positioning system, power inverters, test cables, socket were all well connected to ensure smooth data collection. A digital map of the area was also useful. The transmitter-receiver parameters used for the experiment include: Transmitter frequency- 1900MHz, Transmitter power- 43dB, Transmitter antenna height-34m, Receiver antenna height-1.5m and Receiver antenna gain-18dBi.

ADALINE and MLP Network Training and Prediction

The neural networks: ADALINE and MLP networks were trained using Bayesian Regularization training function (trainbr). This training function is coded in MATLAB and implemented using ANN toolbox (nntraintool) in MATLAB 2013a. One hundred and seventy-six (176) measure data of electromagnetic signal power over different distances were used as inputs to the neural network during the training. To ensure there is no bias in the order of presentation of the data pattern to the ANN, the measure data were normalized in excel spread sheet. Applying Bayesian Regularization approach, 90% of the data were used for training while the remaining 10% were used for validation. The performance metrics used for error analysis after the network training are: the root mean square error (RMSE), standard deviation (SD), mean absolute error (MAE) and the correlation coefficient (r). All these parameter were used to check the error between the actual values and the predicted values using different values of learning rate on the MLP and ADALINE network with ADALINE having an input delay of 1:5. The performance metrics used are defined as [25]:

$$RMSE = \sqrt{\frac{1}{N_{exp}} \sum_{p=1}^{N_{exp}} [l(s) - y_0(n)]^2} \tag{18}$$

$$MAE = \frac{1}{N_{exp}} \sum_{p=1}^{N_{exp}} l(s) - y_0(n) \tag{19}$$

$$SD = \sqrt{(\frac{1}{N_{exp}} \sum_{i=1}^{N_{exp}} | l_s - y_0 | MAE)^2} \tag{20}$$

$$r = \frac{N_{exp} \sum l(s) - y_0(n) - (\sum l(s)(\sum y_o)}{\sqrt{N_{exp}(\sum l(s))^2 - (\sum l(s))^2} \sqrt{N_{exp}(\sum y_0(n))^2 - (\sum y_0(p))^2}} \tag{21}$$

N_{exp} = number of the measured data, $l(s)$ = measured signal power loss values, $y_0(n)$ = the neural network output. A closer correlation coefficient to +1 or -1 shows a high degree of closeness of the predicted values to the measure values.

Table 2. BP ANN model Parameters (MLP)

Parameter	Function/Method/Value
Network Architecture	2-layers
Input data	176
Layer neurons	176-40-1
Data normalization	Excel spread sheet
Transfer function	tansig,logsig
Training function	trainbr

Table 3. ADALINE Parameters

Parameter	Function/Method/Value
Network Architecture	1-layer
Input data	176
Layer neurons	176-1
Data normalization	Excel spread sheet
Transfer function	purelin
Training function	trainbr

RESULTS AND DISCUSSIONS

The statistical performances of ADALINE with input delay of 1:5 and MLP network with different variation of their learning rate are shown in tables 4. Table 5 shows the gradient and momentum values for ADALINE and MLP network at different variation of learning rate parameter. The predictive abilities of the ADALINE and MLP networks were also examined using different combination of non-linear transfer functions mostly used for BP such as logistic sigmoid and hyperbolic tangent and linear transfer functions such as purelin. The performance error with combination of the different transfer functions are shown in table 6 and 7.

Table 4. Training Result of MLP Network and ADALINE with 1:5 input delay

ARTIFICIAL NEURAL NETWORKS PARAMETERS FOR COMPARISON	ADALINE WITH INPUT DELAY (Learning rate %)					MULTI-LAYER PERCEPTRON WITH NO INPUT DELAY (Learning rate %)				
	0.1	0.3	0.5	0.7	0.9	0.1	0.3	0.5	0.7	0.9
Correlation co-efficient (r)	0.9905	0.9910	0.9894	0.9883	0.9877	0.9884	0.9896	0.9907	0.9911	0.9915
RMSE	1.8610	1.8121	1.9660	2.0610	2.1184	2.0537	1.9399	1.8627	1.8055	1.7525
MAE	1.4746	1.3771	1.5270	1.5868	1.6771	1.4473	1.5197	1.3946	1.3722	1.3900
SD	1.1354	1.1778	1.2383	1.3152	1.2941	1.4571	1.2057	1.2348	1.1733	1.0673
Training Time	28	28	30	30	28	29	28	28	28	28
Epoch Iteration (1000)	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

Table 5. Gradient and Momentum parameters for ADALINE and MLP at different learning rate

LEARNING RATE (%)	GRADIENT		MOMENTUM PARAMETER	
	ADALINE	MULTI-LAYER PERCEPTRON	ADALINE	MULTI-LAYER PERCEPTRON
0.1	0.66807	0.31436	0.050	0.005
0.3	0.06772	0.38942	0.050	0.005
0.5	0.69185	0.40904	0.050	0.050
0.7	0.05764	0.42981	0.050	0.050
0.9	0.07441	0.44194	0.050	0.050

Table 7. Statistical errors of the MLP network with different combination of non-linear and linear transfer functions in the hidden and output layer

MULTI-LAYER PERCEPTRON NETWORK				
Transfer Function	**RMSE**	**SD**	**MAE**	**r**
logsig,logsig	2.0244	1.4199	1.4430	0.9887
logsig,tansig	1.9760	1.2377	1.5402	0.9892
tansig, logsig	1.8671	1.1275	1.4882	0.9904
logsig,purelin	2.2009	1.5020	1.6087	0.9866
tansig,purelin	2.1007	1.4353	1.5340	0.9878

Table 8. Statistical error of ADALINE with linear and non-linear transfer functions

ADALINE				
Transfer Function	**RMSE**	**SD**	**MAE**	**r**
logsig	2.1457	1.3746	1.6475	0.9874
tansig	2.3167	1.5182	1.7499	0.9853
purelin	1.8846	1.2139	1.4416	0.9903

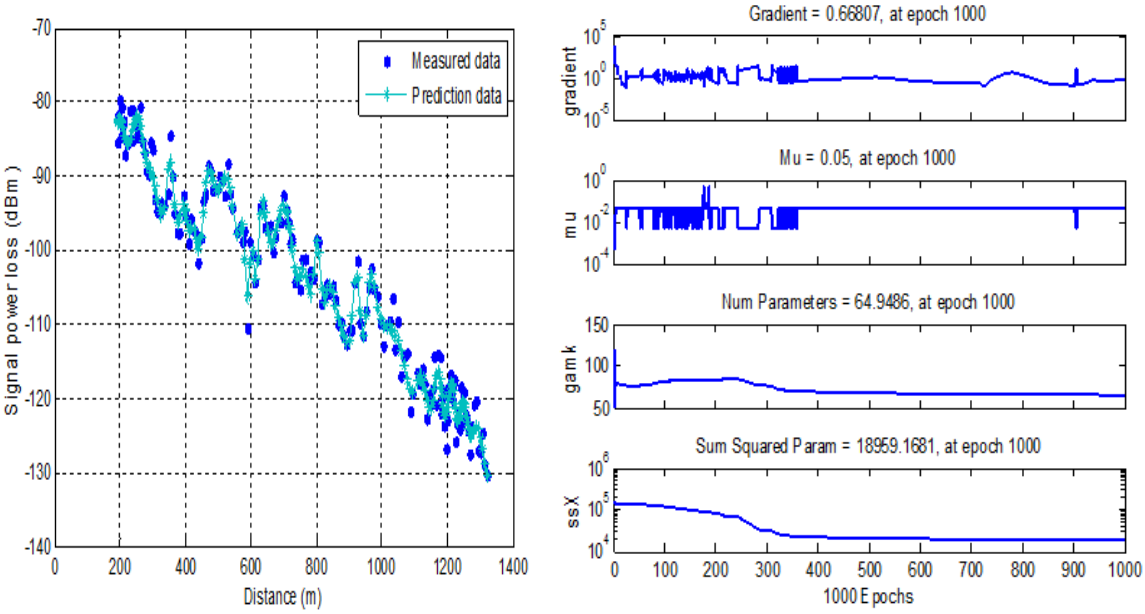

Figure 6. Prediction graph and training state of ADALINE at 0.1% learning rate

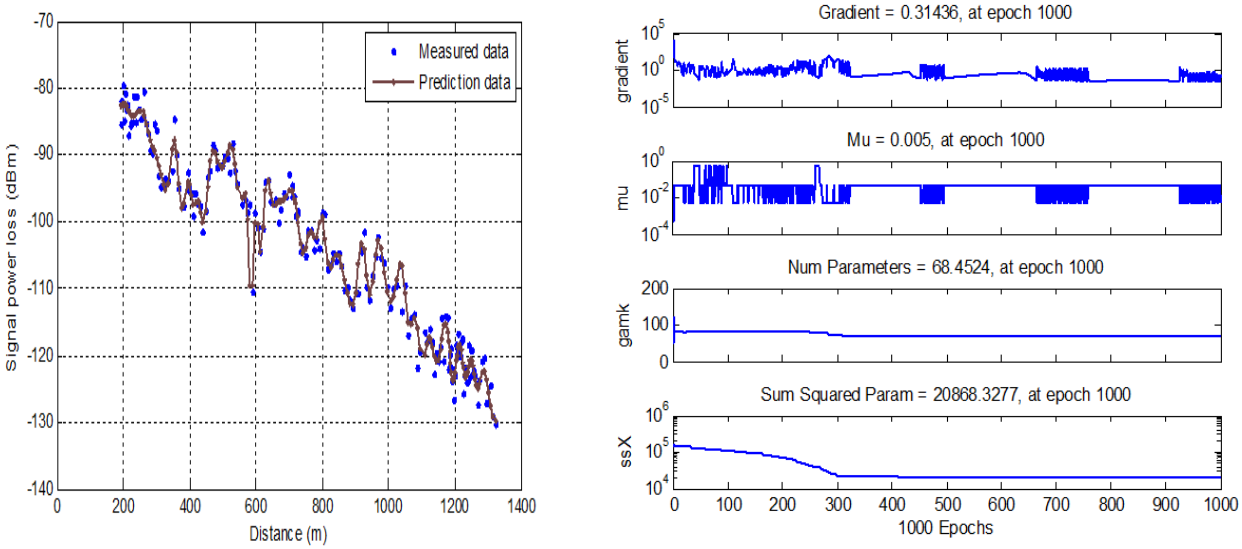

Figure 7. Prediction graph and training state of MLP Network at 0.1% learning rate

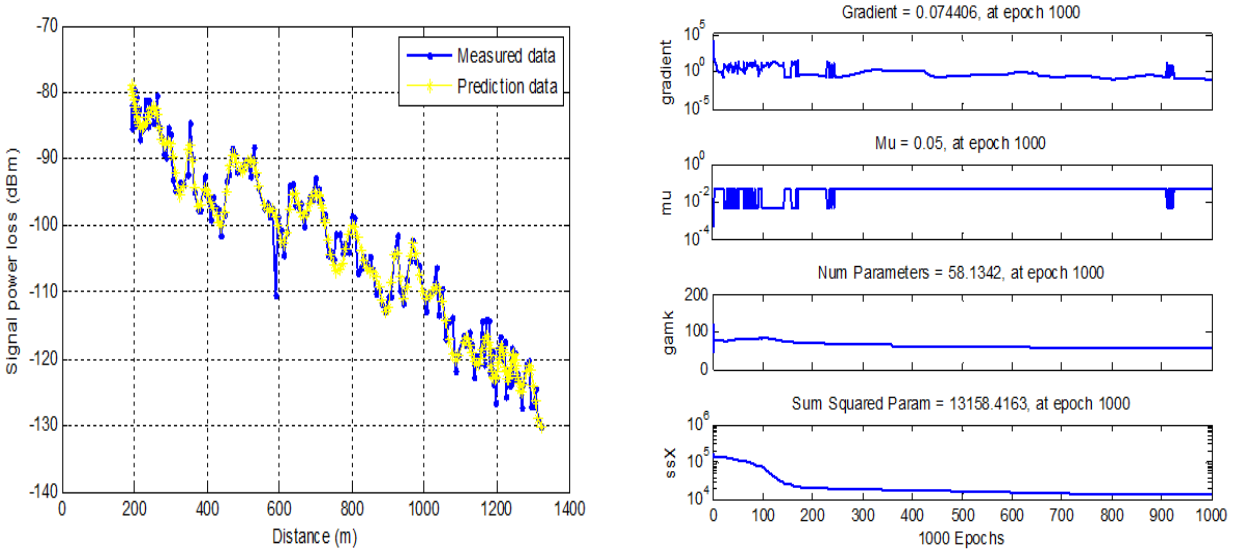

Figure 8. Prediction and training state of ADALINE at 0.9% learning rate

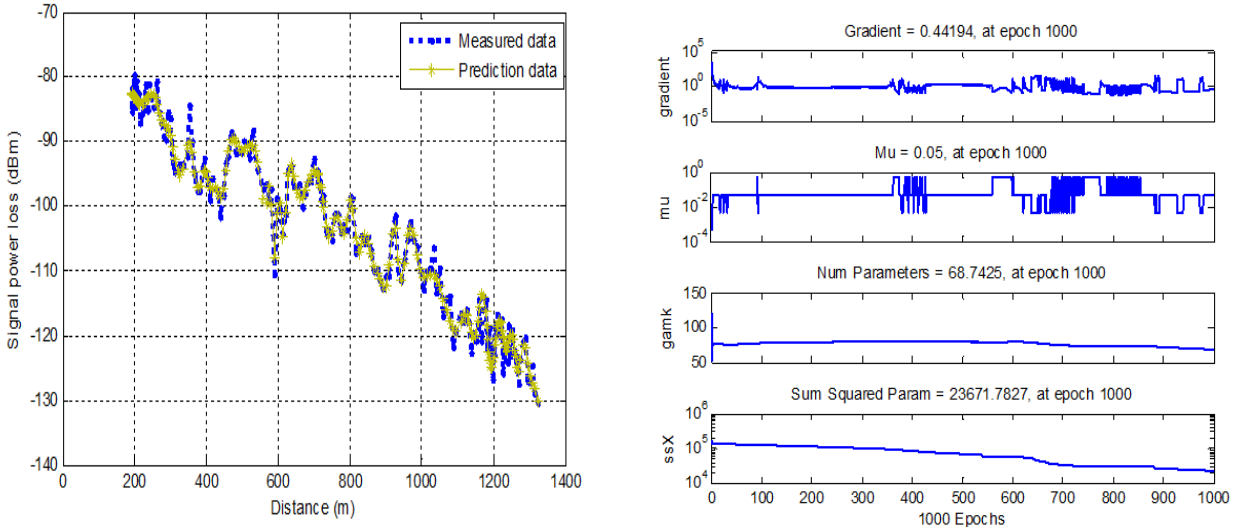

Figure 9. Prediction and training state of MLP Network at 0.9% learning rate

From the statistical performance of the predictive capabilities of ADALINE with input delay and MLP network as shown in table 4 using error measurement of the differences between the actual and the predictive values, a small learning rate of 0.1% with ADALINE network gave a better predictive result than the MLP network, however the gradient was 0.66707 with momentum of 0.05 which is slope movement beyond the optimal and may result to overshoot. At 0.1% learning rate, the gradient of MLP network is 0.31436 with a momentum of 0.005 which is a slope movement towards the optimal i.e. the local minima. Better momentum value as seen with MLP network helps to increase convergence speed of the network. These are shown in figures 6 and 7 respectively. With increase in learning rate to 0.9%, the prediction ability of MLP network became better than that of ADALINE but the gradient increases to 0.44194 with a momentum of 0.05 while the gradient of ADALINE reduces to 0.074406 with a momentum of 0.05. These are shown in figures 8 and 9. Therefore, with increase in learning rate, high gradient parameter as seen in ADALINE will result to faster convergence, however this may also lead to system overshoot. Gradient descent shows an effective use in determining optimal weights by assisting in searching for the optimal value of a cost function. Tables 7 and 8 shows the training performance of the ADALINE and the MLP network with different non-linear and linear transfer functions. Training performance of purelin transfer function in ADALINE shows the least errors because it is a linear transfer function which is most appropriate for a linear neural network such as ADALINE. The combination of different transfer function in the hidden and output layer of MLP network shows the least performance error with the combination of hyperbolic tangent and logistic sigmoid in the hidden and output layers respectively.

CONCLUSION

This work explored the predictive abilities of ADALINE and MLP neural networks in prediction of signal power loss during electromagnetic signal propagation from the transmitter to the receiver. Four measurement error indices: the RMSE, SD, MAE and correlation coefficient were used to ascertain the error differences between the actual measurement from an LTE network and the prediction values. ADALINE with an input delay of 1:5 and a MLP network were trained with learning rate 0f 0.1, 0.3, 0.5, 0.7 and 0.9 and their prediction error, gradient and momentum parameters compared. Also different combinations of linear and non-linear transfer functions were used during the two network training. It was established that ADALINE predicted the measured data better than the MLP network at small learning rate of 0.1%, however there is a positive movement of the gradient away from the local minima while MPL network though with higher prediction error at 0.1% learning rate show a better downward gradient movement towards the slope where the rate of change in function equals zero. The prediction performance of MLP network increased with increase in learning rate of 0.9%, but the gradient moves farther from the local minima while ADALINE has the best gradient of 0.074406 which is approximately at the local minima where the rate of function changes equal zero at 0.9% learning rate. Therefore, the is need for selection of an appropriate learning rate for network training that will be high enough to increase the rate of convergence but not too high to cause system overshoot. Furthermore, training MLP network with hyperbolic tangent and logistic sigmoid in the hidden and output layers shows the least performance error. While training with hyperbolic and purelin and logistic sigmoid and purelin in the hidden and output layers respectively demonstrated that a linear transfer function can effectively be used in the output layer of a MLP network. However, there is need for a non-linear transfer function in the hidden layer to prevent linearly separable solutions. Training ADALINE with purelin gave the best predictive result which is in good agreement as purelin is a linear transfer function adequately use for linear system such as ADALINE.

REFERENCES

[1] D. F. Lekkas, C. Onof, M. J. Lee, and E. A. Baltas, "Application Of Artificial Neural Networks For Flood Forecasting," *The International Journal of Global Nest,* vol. 6, pp. 205-211, 2004.

[2] D. E. Rumelhart, E. Hinton, and J. Williams, "Learning internal representation by error propagation," *Parallel Distributed Processing,* pp. 318-362, 1986.

[3] T. Kohonen, "Self-Organized Formation of Topologically Correct Feature Maps," *Biol. Cybern,* vol. 43, pp. 59-69, 1982.

[4] J. J. Hopfield, "Learning Algorithms and Probability Distributions in Feed-Forward and Feedback Networks," *P. Natl. Acad. Sci. USA,* vol. 84, pp. 8429-8433, 1987.

[5] M. J. D. Powell, *Radial Basis Functions for Multivariable Interpolation: A Review in Algorithms for Approximation* Oxford: Clarendon Press, 1987.

[6] R. Hecht-Nielsen, "Counterpropagation Networks," *Appl. Optics,* vol. 26, pp. 4979-4984, 1987.

[7] J. L. Elman, "Finding structure in time," University of California at San Diego, Centre for Research in Language, California1988.

[8] B. Widrow and M. A. Lehr, "30 years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation," *Proceedings of the IEEE,* vol. 78, pp. 1415-1442, 1990.

[9] R. M. Balabin, R. Z. Safieva, and E. I. Lomakina, "Comparison of linear and nonlinear calibration models based on near infrared (NIR) spectroscopy data for gasoline properties prediction," *Chemometr Intell Lab. ,* vol. 88 pp. 183-188, 2007.

[10] T. B. Ludermir, "Hybrid Optimization Algorithm for the Definition of MLP Neural Network Architectures and Weights," *IEEE Proceedings of the Fifth*

International Conference on Hybrid Intelligent Systems 2005.

[11] J. Isabona and V. Srivastava, M. , "Radio Channel Propagation Characterization and Link Reliability Estimation in Shadowed Suburban Macrocells," *International Journal on Communications Antenna and Propagation* pp. 7-11, 2017.

[12] J. Isabona and V. Srivastava, M., "Coverage and Link Quality trends in Suburban Mobile Broadband HSPA network Environments " *Wireless personal Communication (WPC),* vol. 92, pp. 3955-3968, 2017.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature,* vol. 323, pp. 533-536, 1986.

[14] P. J. Werbos, *The Roots of Backpropagation. From Ordered Derivatives to Neural Networks and Political Forecasting*. New York, NY: John Wiley & Sons, 1994.

[15] S.-S. Shai and B.-D. Shai, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge: Cambridge University Press, 2014.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing :Explorations in the Microstructure of Cognition*, ed: MIT Press 1986, pp. 318-362.

[17] A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Second ed.: SIAM, 2008.

[18] E. Mizutani, S. Dreyfus, and K. Nishio, "On derivation of MLP backpropagation from the Kelley-Bryson optimal-control gradient formula and its application," *Proceedings of the IEEE International Joint Conference on Neural Networks,* 2000.

[19] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional Neural Networks for Speech Recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing,* vol. 22, pp. 1533-1545, 2014.

[20] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks.,* vol. 61, pp. 85-117, 2015.

[21] Y. Li, Y. Fu, H. Li, and S. W. Zhang, "The Improved Training Algorithm of Back Propagation Neural Network with Self-adaptive Learning Rate," *International Conference on Computational Intelligence and Natural Computing,* vol. 1, pp. 73-76, 2009

[22] "Amnesty International Nigeria : Petroleum Pollution and Poverty in the Niger Delta," *United Kingdom: Amnesty International Publications International Secretariat,* p. 10, 2009.

[23] N. J. NNA and B. G. PABON, "Population, Environment and Security in Port-Harcourt," *IOSR Journal of Humanities and Social Science (JHSS),* vol. 2, pp. 01-07, 2012.

[24] "Nigeria's First National Communication Under the United Nations Framework Convention on Climate Change. UNFCC.," 2003.

[25] F. H. Thanoon, "Robust Regression by Least Absolute Deviations Method," *International Journal of Statistics and Applications,* vol. 5, pp. 109-112, 2015.