Modeling Rainfall in Cross River State, Nigeria, Using Artificial Neural Network

Christian E. Onwukwe, Ikpang Nkereuwem Ikpang

Abstract— The Artificial Neural Network approach is utilized in this study for modeling rainfall in Cross River State, Nigeria. The architecture employed for model development is Feed Forward Back-Propagation Architecture [FFBPA]. The model is developed to forecast 12 months rainfall in advance. The study considers three major local government areas in Cross River State; and three separate ANN models were developed using MATLAB for these areas. Rainfall data ranging from 2004 to 2014 were collected from the Nigerian Meteorological Agency [NIMET] and used for training the models based on Gradient Descent Training Algorithm [GDTA]. The effect of number of hidden nodes on the result is examined by varying its numerical value; and with this, the best model was chosen based on Mean Square Error [MSE]. The study showed that the best performance was obtained when the number of nodes in the hidden layer was equal to the number of nodes in the input layer.

Index Terms— Rainfall, Artificial Neural Network, Feed-Forward Neural Network, Back Propagation Architecture Technique, Gradient Descent Training Algorithm, Hyperbolic Tangent Transfer function, Mean Square Error

1 INTRODUCTION

Rainfall forecasting is a complicated procedure that includes specialized fields of know-how [1]. In recent times, climate change and variability constitute serious threat to global socio-economic development and human as well. Climate events such as floods have been on increase in recent time all over the world and this trend has been attributed to global warming and climate change.

Rainfall is a climate parameter that affects every facet of the ecological system, flora and fauna inclusive hence the study of rainfall is important and cannot be over emphasized [2]. In Cross River State, Nigeria, rainfall has profound impact on agriculture, air and land transportation, hydroelectric power generation construction, and water resources, etc.; hence, normal rainfall is beneficial for agriculture and other economic activities. However, when it is excessive or above normal, it may result in flooding and associated negative impacts. Therefore, accurate information on rainfall is essential for planning, though rainfall is one of the most complex and difficult elements of the hydrology cycle to understand and model, due to complexity of atmospheric processes that generate rainfall and tremendous range of variation over a wide range of scales both in space and time [3].

Rainfall is a natural phenomenon that is beyond human control and it is expected that rainfall models should be developed while existing models should be updated as a result of climate change. Furthermore, Box and Jenkins techniques like ARMA models have been used in the past for developing forecasting models in this regard but it requires empirical knowledge of the data used. However, in recent times, neural network approach has been found to overcome this flaw; but has only been successfully used within the country to model rainfall in Akure, Ondo state [4]. In that work, the model parameters were obtained from data collected within the locality and may not work adequately for other parts of the country. In this vain, this paper implements the Neural Network approach to model rainfall in Cross River State, Nigeria.

Rainfall dynamics are dependent upon highly unpredictable physical parameters. These parameters are pressure, temperature, humidity, wind direction, wind speed and cloud amount. Considering these parameters individually make statistical models more complex therefore initial assumptions are usually made about these parameters because of their importance. However, recent development in pattern recognition and artificial intelligence provide solutions [5].

Artificial neural network (ANN), which is based on the neural structure of the human brain and its complex pattern recognition, is devoid of making initial assumptions. Data used is allowed to govern the process by itself through generation of input–output mapping for the set of data, however complex. Training the network with relevant data enables the network the ability to make predictions based on any input it encounters [6] and [7]. In this study rainfall forecast model for Cross River State using artificial neural network has been developed with which rainfall has been forecasted.

2 METHODOLOGY

The major steps in designing the data forecasting model is as follows:

2.1 Choosing Variables and Data Collection

Rainfall data were obtained from the Nigerian meteorological Agency [NIMET] Calabar Cross River State, Nigeria. The data for Calabar, Ikom and Ogoja were collected on a monthly basis for a period of 10 years.

2.2 Data pre-processing

The data collected is preprocessed using the map-minimax transformation code in MATLAB that normalizes the data using the formula:

$$SV = TF_{\min} + \frac{(TF_{\max} - TF_{\min})(X_0 - X_{\min})}{(X_{\max} - X_{\min})}$$

Where

SV	=	Scaled value
$TF_{\min}$	=	Minimum value of transformation function
$TF_{\max}$	=	Maximum value of transformation function
X_0	=	Value of the observation
$X_{\max}$	=	Maximum value of original data set
$X_{\min}$	=	Minimum value of original data set

2.3 Dividing the Data Set into Smaller Sets: Training, Validation and Test

The time series is divided into three distinct sets called the training, validation and testing [out of sample] sets. The training set being the biggest set employed in training the network is 70% of the data while 15% of the data is used as the validation set to evaluate the generalization ability of the trained network while avoiding overtraining the network. Finally the remaining 15% of the data is used as the testing set to check the performance of the trained network.

2.4 Determining Network Topology

This step involves determination of number of input nodes, number of hidden layers and hidden nodes, and the number of output nodes.

2.4.1 Number of Input Nodes

The number of input nodes corresponds to the number of variables in the input vector used to forecast future values. In this study, the number of input nodes corresponds to the number of lagged observation used to discover the underlying pattern in a time series and to make forecast; but there is no suggested systematic way to determine this number. The intuitive or empirical idea by [8] using 12 input nodes for monthly data was adopted.

2.4.2 The Number of Hidden Layer and Hidden Nodes

The hidden layers and nodes play very important roles for many successful applications of neural networks. Several rules of thumb have been proposed for determining the number of hidden nodes like: [9] proposing that the number of node in the hidden layer should range from $(2n+1)$ to $(\sqrt[2]{n} + m)$, Sharda and Patil [8] reporting that best results are obtained when number of hidden nodes is equal to number of input nodes etc.; but we have adopted the most common method which is by experimentation or by trial and error using one hidden layer while varying the number of hidden nodes in this layer.

2.4.3 Number of Output Nodes

The number of output mode corresponds to the forecasting horizon; hence, we will have one output node in this research

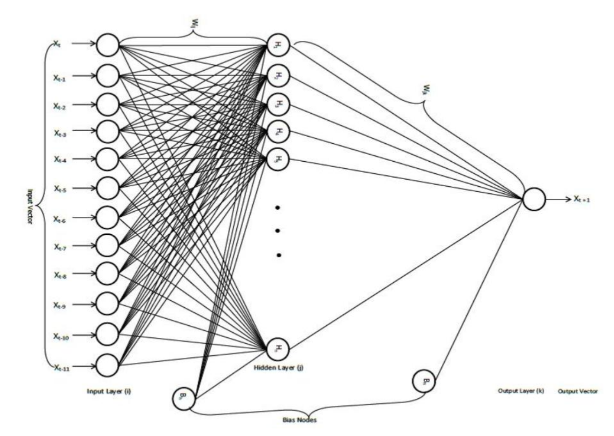

Fig. 1.Schematic representation of the ANN architecture

2.5 Training

This study adopts gradient descent back propagation algorithm training method for training the network. This algorithm involves two major phases; a feed forward phase (in which external input information at the input nodes is propagated forward to compute the output information) and the backward phase (in which modifications to the connection strength are made based on differences between desired and actual output).

First, we feed input entry data into the network through the input layer neurons. To compute temporary outputs, the product of the input values and the first interconnection arc weight $w_{ij}, j = 1,...,h$ initialized by assigning random values, are computed. The product of the input values and these weights at the hidden nodes are summed over the index i and associates it with a bias (B) to produce the inputs to the hidden layers. That is,

$$H_j = \sum_{i=1}^{n} w_{ij} x_i + B_h \qquad (1)$$

where H_j is the input to the j^{th} hidden node, W_{ij} is the arc weight connecting the i^{th} processing element of the input layer to the j^{th} processing element of the hidden layer and B is the bias term usually assigned the value one (1).

Input to each hidden node is transformed through an activation function to generate a hidden node output using the sigmoid function because it is continuous and differentiable. The hyperbolic tangent function is adopted in this work and defined as

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad (2)$$

So that $f'(x) = (1 - f(x)^2)$

The output, denoting the transformed hidden node sum in the hidden layer, function as inputs to the successive layer, which in this case, is the output layer. The same operation is carried out on the neurons preceding the output layer so that,

$$H_o = \sum w_{jk}\ f_h\left(B_h + \sum_{i=1}^{n} w_{ij} x_i\right) + B_o \qquad (3)$$

where H_0 is the input to the o^{th} output node and w_{jk} is the arc weight connecting the j^{th} processing element of the hidden layer to the k^{th} processing element of the output layer and B_o is also the bias term.

The entire process can therefore be written mathematically as

$$O_k = f_0\left\{B_0 + \sum_{j=1}^{n} w_{jk}\left(f_h\left(B_h + \sum_{i=1}^{n} w_{ij} x_i\right)\right)\right\} \qquad (4)$$

We let the error of the network for a single training interaction to be denoted by

$$E = \frac{1}{2}\sum (o_k - t_k)^2 \qquad (5)$$

where o_k is the predicted output from the network and t_k is the actual or targeted value of the input values. We calculate the rate of change of the error with respect to the given connective weight, so we can minimize it or do a gradient descent on the gradient of the error, with respect to the weights and adjust it. We have considered two cases for this namely: the node is an output node and the node is in a hidden layer.

1. The node is an output node

$$\frac{\partial E}{\partial w_{jk}} = \frac{\partial}{\partial w_{jk}} \frac{1}{2}\sum_{k\in K}(o_k - t_k)^2 \qquad (6)$$

"where j = element of hidden layer, k = element of output layer".

By chain rule

$$\frac{\partial E}{\partial w_{jk}} = (o_k - t_k)\frac{\partial}{\partial w_{jk}} o_k$$

But $o_k = f_0(x_k)$

$$\frac{\partial E}{\partial w_{jk}} = (o_k - t_k)\frac{\partial}{\partial w_{jk}} f_0(x_k)$$

$$\frac{\partial E}{\partial w_{jk}} = (o_k - t_k)\left(1 - f_0(x)^2\right)\frac{\partial}{\partial w_{jk}} x_k$$

x_k is the output of the j^{th} node multiplied by the connection weight from j to k.

We recall that

$o_k = f_0(x_k)$ and $\frac{\partial}{\partial w_{jk}} x_k = o_j$

Then

$$\frac{\partial E}{\partial w_{jk}} = (o_k - t_k)\left(1 - o_k^{\,2}\right) o_j \qquad (7)$$

Let $\delta_k = (o_k - t_k)\left(1 - o_k^{\,2}\right)$ then we can rewrite the equation as

$$\frac{\partial E}{\partial w_{jk}} = o_j \delta_k \qquad (8)$$

2. The node is in a hidden layer

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \frac{1}{2}\sum_{k\in K}(o_k - t_k)^2 \qquad (9)$$

Where i is the element of the input layer and j is the element of the hidden layer.

$$\frac{\partial E}{\partial w_{ij}} = \sum_{k\in K}(o_k - t_k)\frac{\partial}{\partial w_{ij}} o_k$$

Again $o_k = f_0(x_k)$

$$\frac{\partial E}{\partial w_{ij}} = \sum_{k\in K}(o_k - t_k)\left(1 - f_0(x)^2\right)\frac{\partial}{\partial w_{ij}} x_k$$

$$\frac{\partial E}{\partial w_{ij}} = \sum_{k\in K}(o_k - t_k)\left(1 - o_k^{\,2}\right)\frac{\partial x_k}{\partial o_j}\cdot\frac{\partial o_j}{\partial w_{ij}}$$

But $\frac{\partial x_k}{\partial o_j} = w_{jk}$

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial o_j}{\partial w_{ij}}\sum_{k\in K}(o_k - t_k)\left(1 - o_k^{\,2}\right) w_{jk}$$

$$\frac{\partial E}{\partial w_{ij}} = \left(1 - o_j^{\,2}\right)\frac{\partial x_j}{\partial w_{ij}}\sum_{k\in K}(o_k - t_k)\left(1 - o_k^{\,2}\right) w_{jk}$$

x_j, the input to j, is the output of the i^{th} node multiplied by the connection weight from i to j

$$\frac{\partial E}{\partial w_{ij}} = \left(1 - o_j^{\,2}\right) o_i \sum_{k\in K}(o_k - t_k)\left(1 - o_k^{\,2}\right) w_{jk}$$

Recall that $\delta_k = (o_k - t_k)\left(1 - o_k^{\,2}\right) \qquad (10)$

$$\frac{\partial E}{\partial w_{ij}} = o_i\left(1 - o_j^{\,2}\right)\sum_{k\in K}\partial_k w_{jk} \qquad (11)$$

Let $\delta_j = \left(1 - o_j^{\,2}\right)\sum_{k\in K}\partial_k w_{jk}$

Then

$$\frac{\partial E}{\partial w_{ij}} = o_i \delta_j \qquad (12)$$

If the bias term incorporated, we have

$$\frac{\partial O}{\partial B} = 1$$

So the bias term is viewed as output from a node which is always one (1) and holds for any layer and the back propagation algorithm therefore is as follows:

Given the set of input data,

i. Initialize all weights and bias and transform the training data.
ii. Run the network forward with the input data to get network output using
$$O_k = f_0\left\{B_0 + \sum_{j=1}^{n} w_{jk}\left(f_h\left(B_h + \sum_{i=1}^{n} w_{ij} x_i\right)\right)\right\}$$
iii. For output node, compute
$$\delta_k = (o_k - t_k)\left(1 - o_k^{\,2}\right)$$

iv. For each hidden node, calculate

$$\delta_j = (1 - o_j^2) \sum_{k \in K} \partial_k w_{jk}$$

v. Update weight and biases as follows:

$$\Delta w_t = -\eta \frac{\partial E}{\partial w_t} + \alpha \Delta w_{t-1}$$

$$\Delta B = -\eta \frac{\partial E}{\partial w_t}$$

And apply

$$w_t = w_t + \Delta w_t$$
$$B = B + \Delta B$$

Where

w_t	=	weight concerned with
η	=	learning rate
$\propto$	=	momentum
Δw_{t-1}	=	previous change in weight at node
$\frac{\partial E}{\partial w_t}$	=	gradient at node concerned with. That is,

$\frac{\partial E}{\partial w_{jk}} = o_j \delta_k$ if the node in an output node or

$\frac{\partial E}{\partial w_{ij}} = o_i \delta_j$ if the node is in the hidden layer.

The process is repeated from step two (2) after weights update until the error converges to an acceptable level.

3 IMPLEMENTATION

In this study, MATLAB was used for implementing the network. Three models were developed for Calabar, Ikom and Ogoja respectively since rainfall intensity differ for these areas in Cross River State, Nigeria. The interconnection weights were initialized randomly taking values between 0 and 1while, maximum number of epochs in this study is set to 1000 epochs. More than 20 experiments were performed to determine the best combination of learning rate, momentum, transfer function and training function. The hyperbolic tangent sigmoid transfer function using gradient descent back propagation algorithm was used. The performance of the network is measured in terms of its Mean Square Error and correlation and hence, forms the basis for choosing the best model.

Initially, an ANN with five nodes in the hidden layer is defined while the MSE and correlation of the output and target is obtained for this case. This hidden node value was varied using ten, twelve, fifteen, twenty, twenty-five and thirty and for each case, the network is trained and retrained for best result.

Table 1 evidently shows that the best result was with 12 neurons in the hidden layer having the least MSE value of 0.135. Figure 2 shows the performance plot [plot of training, validation and testing errors] for Calabar. From this plot, we observe that the validation error curve and test error curve are very similar hence no major problems with the training. The best architecture for the Calabar rainfall data set with minimum corresponding MSE is a 12-12-1 [12 input nodes, 12 hidden layer nodes and 1 output node] model.

TABLE 1
PERFORMANCE EVALUATION BASED ON HIDDEN LAYER SIZE FOR CALABAR

Hidden Layer Nodes	Epoch	MSE	R
5	1000	0.139	0.797
10	1000	0.084	0.677
12	1000	0.135	0.789
15	1000	0.216	0.77
20	1000	0.142	0.809
25	1000	0.191	0.76
30	1000	0.133	0.777

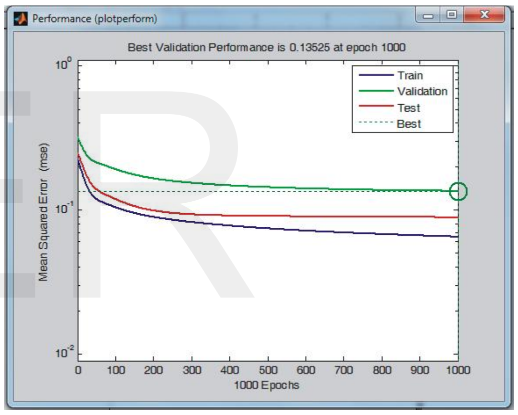

Fig. 2. Snapshot of performance plot for rainfall data of Calabar

Table 2 evidently shows that the best result was with 12 neurons in the hidden layer having the least MSE value of 0.064. Figure 3 shows the performance plot [plot of training, validation and testing errors] for Ikom. From this plot, we observe that the validation error curve and test error curve are very similar hence no major problems with the training. The best architecture for the Ikom rainfall data set with minimum corresponding MSE is a 12-12-1 [12 input nodes, 12 hidden layer nodes and 1 output node] model.

TABLE 2
PERFORMANCE EVALUATION BASED ON HIDDEN LAYER SIZE FOR IKOM

Hidden Layer Nodes	Epoch	MSE	R
5	1000	0.084	0.782
10	1000	0.139	0.762
12	1000	0.064	0.835
15	1000	0.108	0.831
20	1000	0.095	0.825
25	1000	0.097	0.833
30	1000	0.153	0.800

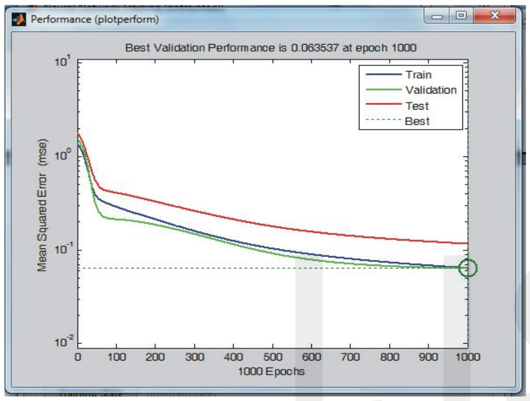

Fig. 3. Snapshot of performance plot for rainfall data of Ikom

Table 3 evidently shows that the best result was with 12 neurons in the hidden layer having the least MSE value of 0.18667. Figure 4 shows the performance plot [plot of training, validation and testing errors] for Ogoja. From this plot, we observe that the validation error curve and test error curve are very similar hence no major problems with the training. The best architecture for the Ogoja rainfall data set with minimum corresponding MSE is a 12-12-1 [12 input nodes, 12 hidden layer nodes and 1 output node] model.

TABLE 3
PERFORMANCE EVALUATION BASED ON HIDDEN LAYER SIZE FOR OGOJA

Hidden Layer Nodes	Epoch	MSE	R
5	1000	0.200	0.782
10	1000	0.202	0.833
12	1000	0.187	0.823
15	1000	0.280	0.728
20	1000	0.242	0.702
25	1000	0.213	0.793
30	1000	0.197	0.781

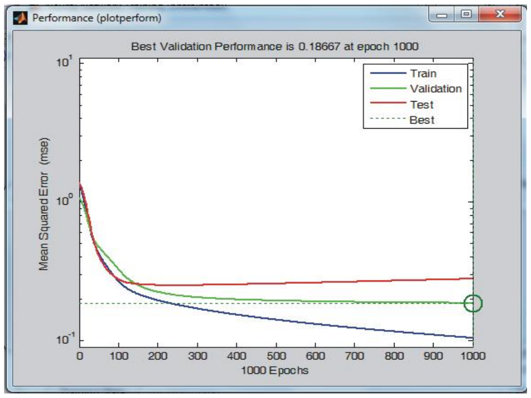

Fig. 4. Snapshot of performance plot for rainfall data of Ogoja

4 FORECAST

According to results obtained, the back propagation neural networks were acceptably accurate and were used for forecasting rainfall in Cross River State, Nigeria. Forecasts are as shown in Table 4, Table 5 and Table 6.

TABLE 4
RAINFALL FORECAST VALUES FOR CALABAR 2015

Month	Rainfall (mm)
January	57.4
February	107.66
March	167.98
April	223.48
May	354.81
June	472.18
July	526.04
August	447.99
September	397.16
October	302.9
November	238.43
December	51.23

TABLE 5
RAINFALL FORECAST VALUES FOR IKOM 2015

Month	Rainfall (mm)
January	18.44
February	56.97
March	84.44
April	176.17
May	252.01
June	326.98
July	324.09
August	333.5
September	279.87
October	231.31
November	73.04
December	9.44

TABLE 6
RAINFALL FORECAST VALUES FOR OGOJA 2015

Month	Rainfall (mm)
January	13.23
February	24.93
March	52.1
April	181.33
May	392.26
June	447.33
July	422.37
August	387.33
September	357.33
October	258.96
November	71.68
December	20.75

4 CONCLUSION

A multilayered feed forward neural network has been developed for three local government areas in Cross River State, Nigeria. To determine the best neural network forecast model, the number of hidden layer nodes was varied and it was observed that for each local government area the best forecast model was obtained when the number of nodes in the hidden layer is equal to the number of nodes in the input layer.

REFERENCES

[1] P. Guhathakurta, "Long range Monsoon rainfall prediction of 2005 for the districts and sub-division Kerala with artificial neural network". *Current science* 90(6): 773-779. 2006.

[2] N. Obot and N.O. Onyeukwu, "Trend of Rainfall in Abeokuta, Ogun State Nigeria: A 2 year experience [2006-2007]" *Journal of Environmental Issues and Agriculture in Developing Countries,* 2 (1): 70 – 81. 2010.

[3] M.N. French, W.F. Krajewski and R.R. Cuykendall, "Rainfall Forecasting in Space and Time using Neural Networks, *Journal of Hydrology*, 137: 1 – 31. 1992.

[4] A.H. Agboola, O. Iyare and S.O. Falaki, "An Artificial Neural Network Model for Rainfall Forecasting in South Western Nigeria". *Canadian Journal on Computing in Mathematics, Natural Sciences Engineering and Medicine*, 1(6), 188-196. 2012.

[5] A.D. Kumarasiri and D.U. Sonnadara, "Rainfall Forecasting: An Artificial Neural Network Approach. *Proceedings of the Technical Sessions*. 22(2006): 1-16. 2006

[6] K.C. Luk, J.E. Ball and A. Sharma, "An Application of Artificial Neural Network for Rainfall Forecasting". *Mathematical and Computer Modeling*, 33: 683 – 693. 2001.

[7] W.S. Philip and K.B. Joseph, "A Neural Network Tool for Analyzing Trends in Rainfalls". *Computers and Geosciences*, 29(2): 215 – 223. 2002.

[8] R. Sharda and R.B. Patil "Connectionist Approach to Time Series Prediction: An Empirical Test" *Journal of Intelligent Manufacturing*, 3 (5): 317 – 323. 1992.

[9] D. Fletcher, and E. Goss, "Forecasting with Neural Networks - An Application using Bankruptcy Data". *Information Management,* 2(24), 159-167. 1993.