International Journal of Applied Engineering Research ISSN 0973-4562 Volume 14, Number 6 (2019) pp. 1305-1312

Design and Analysis of Adaptive Artificial Learning Architecture to Model Non-Linear Systems using Different Activation Functions

R.Rathinasabapathy
Associate Professor,Department of Computer Applications,
Madurai Kamaraj University, Madurai, India.

M.Ravindran
Associate Professor,PG department of computer Science,
Govt. Arts College, Melur, India.

Abstract

The neural networks have also been used to predict the value of an attribute from other attribute values. In this research neural network architectures with no hidden layer and a hidden layer have been proposed to model a non-linear system for the dataset taken. The neural network architectures used various activation functions. The dataset taken for experiment, is split into three different ways and the algorithms were applied on the datasets. The RMSE values have been compared among all the methods. There are differences in the RMSE values obtained from different neural network architectures. For large size of the dataset taken, an artificial neural network(ANN) introduced in this research work with a different activation function is showing less root mean square error value over the other architectures both for training and testing. The experiments used here with different artificial neural networks can be extended to be applied on other data sets and their RMSE values can be compared with the existing neural networks

Keywords: artificial neural networks, activation functions and back-propagation algorithm, gradient descent

I. INTRODUCTION

Neural networks have been widely used in science, technological and commercial applications. It is being used for classification, prediction, regression, pattern analysis in data mining [1]-[2]. It is also used in image processing applications such as classification, image identification, and character recognition. It has contributed largely for rain fall prediction, to predict student performance, stock market price prediction and analysis and time series forecasting. The neural networks bring better results than the persistent and conventional methods using numerical and statistical approach[3]. The neural networks are dependent on the neural network architecture, intermediate processing, number of hidden layers, learning rate, activation function used, initial weights, formula to update the weights, methods used to properly train the neural networks to obtain accurate results and the data set for training [4][5][6]. The training set must include all possible cases of the data set that is being trained. The neural networks are hampered by large computing time, not converging to the required results due to incorrect initial weights, inappropriate learning rates, difficulty in finding appropriate activation function, inappropriate neural network architectures [7]. There have been attempts to address these issues in many research articles. Initial weights and learning rates are found using trial and error methods. The learning rate may also have to be modified when the output is oscillating between some maximum and minimum values. Activation functions are chosen by examining the lower and upper bound obtained by these functions and properties of the activation functions. The neural network architecture is to be chosen by examining the nature of the problem to be solved.

II. GENERAL CONCEPTS OF NEURAL NETWORKS

Neural networks[8][9] consist of set of layers of neurons and each layer of neurons are activated by previous layer of neurons. These neurons also activate next layer of neurons and this arrangement is extended until arriving at output neurons. Neurons are the basic units by which the network has been arranged in layers. The first layer is the set of input vectors. These vectors energized by weights to create a set of next layer of neurons. The neural network has been depicted as set of links from one layer to another layer of neuron. Each neuron is attached with the signal which is a linear combination of previous layer of neurons with weights and a bias. These are transformed by an activation function. There are several activation functions being usually used. These activation function values are used to find next layer of neurons in the same way as specified before. The output neuron after transformation by the activation function gives the output of the neural network.

Usually the activation function is nonlinear function. So the neural networks can be viewed as modeling a non-linear system,

$y = f(x_1,x_2,x_3, \ldots x_n)$, for a set of inputs $x_1,x_2,x_3,\ldots x_n$ and the output y.

So it can be considered as a non-linear regression model [10]. In feed forward networks, the signals are transmitted from input vectors (known as first layer of neurons) to second layer of neurons. The second layers of neurons send signals to third

layer of neurons. And this arrangement is extended until arriving at a final output neurons. If the input layers of neurons give rise to output neurons then they are called single feed forward neural networks. Otherwise it is called multi-layer feed forward networks. Using back propagation algorithm, the errors from the output can be reduced. The errors are reduced by updating the weights and other parameters involved in the neural networks. The back propagation algorithms are used in lot of neural network algorithms.

III. DIFFERENT NEURAL NETWORKS AS USED IN VARIOUS RESEARCH WORKS

Neural networks have been used widely in many research works such as classification, clustering, pattern recognition, and prediction of air pollutant concentration [11]. Various activation functions and their applicability for different problems have been discussed in detail in [12]. Rain fall prediction is a most challenging research endeavor. While learning the ANN using Back Propagation algorithm to predict rain fall, it has been observed that multilayer algorithms are better than single layer algorithms and as number of neurons increases the error decreases[13]. ANN model for forecasting rainfall has been obtained using real time data by [14]. It has been observed that the feed forward network with hyperbolic tangent transfer function achieved better results. The ANN model is compared with conventional model and ANN model gives superior results over the conventional method of processing. While learning the network, the error is multiplied with a step size. The step size has to be appropriately chosen, otherwise, it may diverge or it may take too long a time to train. Neural networks are also employed in misuse detection by comparing current activity with the expected actions of an intruder in computer networks[15]. Deep learning techniques are being used to analyze the health care data and found it is more effective than the multiple linear regression methods to predict the outcome of the health care data[16].

IV. ACTIVATION FUNCTIONS AS USED IN OTHER NEURAL NETWORKS

The activation functions are contributing in large extent in the convergence of the learning algorithm. There are several activation functions in use. The activation functions are defined [17] as a class of functions as described below.

f : R→R and takes on values between a and b where a and b belong to R.

$\lim_{x\to-\infty} f(x) = a$ and $\lim_{x\to\infty} f(x) = b$. Usually the value of a is taken as 0 or -1 and b is taken as 1.

The following are sigmoid functions satisfying the conditions specified above. They are unipolar sigmoid, and tanh functions.

The uni-polar sigmoid function is defined as

$f(x)=1/(1+e^{-x})$ and is defined over the interval $(-\infty, \infty)$ and takes values in the interval (0,1).

Hyperbolic tangent function is a defined as

$f(x) = (e^x - e^{-x})/(e^x + e^{-x})$ and is defined over the range $(-\infty,\infty)$ and takes on values (-1,1).

Many of the research literatures use either sigmoid function or hyperbolic tangent function as activation functions, for their neural networks. But there are several other activation functions that have been proposed and have been demonstrated their applicability in various data sets. The activation function hyperbolic secant function[18] has been found to be very effective in classification problems and in multilayer perceptron architectures employing different datasets. The function is said to have the required properties of an activation function such as totally differentiable, symmetric about origin and having finite values when the domain values reach the positive or negative infinite values. Apart from using usual sigmoid functions as activation functions, there are other functions have been used as activation functions in different artificial neural networks employed in different data sets in various research endeavors [19][20]. Also different activation functions have also been evaluated for their performance. Some neural networks[21] are changing the activation function in each layer of the neural network as different activation functions change the results produced.

V. ALGORITHM FOR LEARNING THE NEURAL NETWORK

The neural networks are composed of basic elements known as neurons. The value of a Neuron is found as weighted sum of input vectors or intermediate neuron values. The weights are initialized with appropriate arbitrary values. Each neuron is computed as,

$\sum w_i.x_i + b_i$, Where w_i's are known as weights and b is known as bias.

Usually the neurons are transformed using activation function ø as

$f_i=ø_k(\sum w_i.x_i + b_i)$.

If there is no hidden layer, fi will be the value of the output neurons. Otherwise again the weighted sum of just now computed neurons are found as,

$\sum fj\ wj+bj$

It will be transformed using an activation function as,

$g_q = ø_m(\sum fj\ wj+bj)$,

If it is not the output neuron, then the process is continued in the same manner.

The error in the computed value is given by

$E=\sum(d_i-g_i)^2$, where d_i is the desired value provided in the dataset.

The errors are to be minimized by updating the weights and bias. The weights wi's are to be changed in such a way that the errors are minimized as far as possible. In back propagation algorithms, the weights are updated using the gradient descent of the error function with respect to the corresponding weights. The partial derivative of the error is found to update the weights. The weights are updated with the negative value of the parital derivative multiplied by the learning rate [22].

$w(k+1) = w(k) - \eta.\partial E/\partial w$, where η is the learning rate and w is the weight or a bias.

From the error equation E , it becomes

$w(k+1)=w(k)+\eta.\delta.(d_i-g_i)$ where $\delta=\partial ø_m/\partial w$

The learning rate η is taken to be a sufficiently small and is fixed by trial and error method.

In a multi-layer feed forward networks, there is one or more hidden layers in the neural network architecture. The error in the output has to be back propagated to the previous layers of neurons by updating the weights.

The initial weights are appropriately chosen so that it can converge to the required result. Improper initial weights will not converge to the solution. In this research work we have used neural network architectures using a hidden layer and no hidden layer. We have used three architectures using no hidden layer and one neural network architecture using a hidden layer in this work as detailed in the following chapters.

The following is the algorithm to train the networks in this research work

Begin of Algorithm1

for each iteration

for each vector x_j

//Find weighted sum of the vectors

$\sum w_i.x_i + b$

//Use activation function to transform the weighted

//sum

$f_i = ø(\sum w_i.x_i + b)$

//Find the error in the computation

$error_j = (d_i-f_i)^2$

//Update the weights using gradient descent

$w(k+1) \leftarrow w(k)+\eta.\delta.(d_i-f_i)$, where $\delta=\partial ø/\partial w$

end for

end for.

for each input vector

//find value of fi

$f_i = ø(\sum w_i.x_i + b)$

//find square of the error

$error = (di-fi)^2$

//find the total error

total_error = total_error+error

end for

//Find root mean square error

rmse←sqrt(total_error/number_of_input_vectors)

End of algorithm1

By changing the activation functions, we can find several neural network model.

The following is the algorithm for neural network having one hidden layer. The hidden layer uses two neurons

Begin of algorithm2

For each iteration

For each vector x_j

//Find weighted sum of the input vectors j for first

//hidden neuron

$y_{j1} = \sum w_i.x_i + b_{i1}$

//Find weighted sum of the input vectors j for the //second neuron

$y_{j2} = \sum u_i.x_i + b_{i2}$

//Apply activation functions for the weighted sum

$f_{j1} = ø_1(y_{j1})$

$f_{j2} = ø_2(y_{j2})$

//Find the weighted sum of the hidden layer of neuros

$y_{j3} = \sum q_k.ø_k(y_{jk}) + bi$

//Apply activation function to the weighted sum to //obtain value of output neuron

$f_j = ø\ (y_{j3})$

//Find error in the output

$error_j=(d_j-f_j)^2$

//Update the weights of output layer of neuron using //gradient descent

$q_k=q_k+\eta.\ \delta\ .\ (di-fi)$, where $\delta =\partial ø\ /\partial q_k$

//Update the weights of input layer

$w_i=w_i+\eta.\delta.(di-fi)$ where $\delta = \partial ø/\partial w_i$

$u_i=u_i+\ \eta.\ \delta.(di-f_j)$, where $\delta= \partial ø/\partial u_i$

end for

end for

for each input vector

//find value

error = $(d_j-f_j)^2$

total_error = total_error+error

end for

//Find root mean square error

rmse←sqrt(total_error/number_of_input_vectors)

end of algorithm2

f_{j1} and f_{j2} are hidden layer neuron values.

f_j is the output value which is the predicted value.

$ø_1$ and $ø_2$ are hidden layer activation functions.

ø is the output layer activation function

In the second algorithm, the neural network architecture uses one hidden layer and one output layer. The hidden layer uses an activation function $1/(1+e^{-x})$ and the output layer is using a slightly different activation function $a/(1+e^{-x})$.

VI. NEURAL NETWORKS USED FOR PREDICTION

We have used a feed forward neural network to find the attribute value using other independent attribute values. Back propagation algorithm is used to update the weights by propagating the error found in the feed forward network. The weights are updated using gradient descent as specified in the previous chapter. We have defined three neural networks with no hidden layer and one neural network with one hidden layer. The following are the different activation functions used in the neural networks which are used to train the networks. The activation functions are shown in Table 1.

Table 1: Activation functions used

	Activation function
1	1/(1+exp(-x))
2	a/(1+exp(-x))
3	M/(1+exp(-x))
4	M-M/(1+a.x^2)

The neural networks used in this research work is the following

(i) Neural network with no hidden layer and with an activation function shown in 2 of table 1 and it is denoted as NN1

(ii) Neural network with no hidden layer and with an activation function shown in 3 of table 1 and it is denoted as NN2

(iii) Neural network with no hidden layer and with an activation function shown in 4 of table 1and it is denoted as NN3

(iv) Neural network with one hidden layer. Hidden layer uses the activation function shown in 1 of table 1 and the outer layer uses the activation function shown in 2 of table 1 and it is denoted as NN4

Usually, we will take sigmoid function which will have values in the range (0,1) or (-1,1). The dataset will also be normalized between 0 and 1. But, here we are not normalizing the values, so we need a function which have values in the interval (0, M) where M is the maximum obtainable value of the predicted attribute. Hence we choose activation functions as specified in table 1 and they will lie between intervals as specified below.

The range of activation function shown in 1 of table 1 can be found to be (0,1), shown in 2 of table 1 can be found to be (0,a), shown in 3 of table 1 can be found to be (0,M). The following discussion will elucidate the range of values taken by the activation function specified in 4 of table 1.

Consider the activation function M-M/(1+ax^2). We are fixing the value of a as positive.

Let us explore the range values of the activation function when a>0.

When a>0 and x>=0,

$1+ax^2 \geq 1$ and $1/(1+ax^2) \leq 1$. Hence $M/(1+ax^2) \leq M$.

The values obtainable by $M/(1+ax^2)$, is M when x approaches 0 and is 0 when x approaches infinity.

So M- M/(1+ax^2) will have value 0 when x approaches 0 and will have value M when x approaches infinity.

When a> 0 and x<0, again we have,

$1+ax^2 \geq 1$ and $1/(1+ax^2) \leq 1$. Hence $M/(1+ax^2) \leq M$.

The range of values obtainable by M/(1+ax^2), is 0 when x approaches -∞ (minus infinity) and is M when x approaches 0. So, M - M/(1+ax^2) will have value M when x approaches minus infinity and will have value 0 when x approaches 0. Hence, while learning the neural network using the activation function 4 of Table 1, we fix the value of a as greater than zero.

In all the neural networks the weights and bias are updated as specified in the previous chapters. Apart from weights and bias, we have used other parameters *a* and *M* in the activation functions. While learning the neural networks, the parameter *a* is changed as specified above for NN1. For NN2 and NN3, it has been planned to fix the value of M as maximum value obtainable by the attribute which is to be predicted. The value of *M* is not updated throughout the learning process. In this way the neural networks NN1,NN2, NN3 and NN4 are progressed to model the system. Initially the weights are taken to be minimum and as the learning progresses, the weights are updated. The value of *a* is also provided with initial value and for these neural networks it is taken as 1. In the neural network NN4, the output layer neuron is found using weighted sum of hidden layer neurons and then by applying the activation function.

The Abalone data set provided in UCI machine learning repository dataset has been taken for our experiment and this dataset [23](Asuncion, A. and Newman, D.J., 2007) contains more than four thousand records and we are taking only four

thousand records for our experiment. There are totally nine attributes and eight attributes are numeric values and there is one non-numeric attribute. We are only considering the numeric attributes for our training. We are taking the last attribute as the dependent attribute and other attributes are taken as independent attribute. So there are seven independent attributes and one dependent attribute. The data set for neural networks are taken for experiment in three different ways and the resulting RMSE values of training data and testing data has been tabulated here. In the first case, first 2000 records are used, second 2001 to 4000 records are used in the third case the records 1 to 4000 is used to train and test the networks. In each set sixty percent of data is used to train the dataset and remaining forty percent is used to test the data. The neural networks are named as NNX_Y, where X stands for values 1 through 4 to represent the network and Y takes values A, B and C to represent three types of data set taken for training and testing.

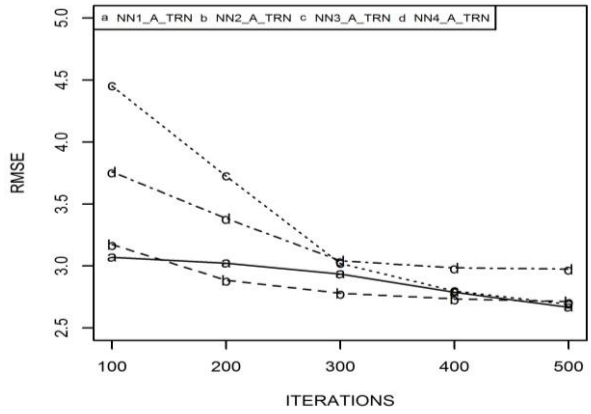

Figure 1: RMSE values obtained while training dataset A

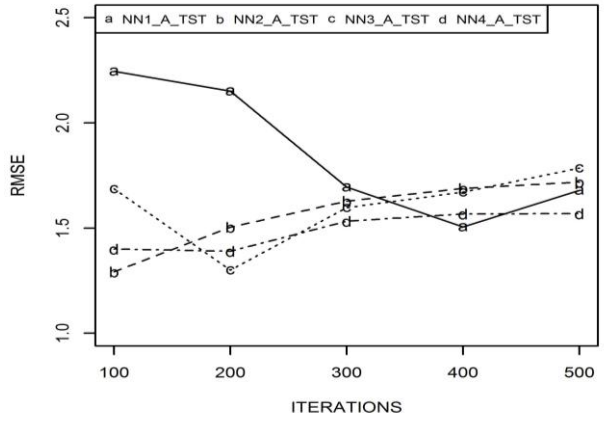

Figure 2: RMSE values obtained while testing dataset A

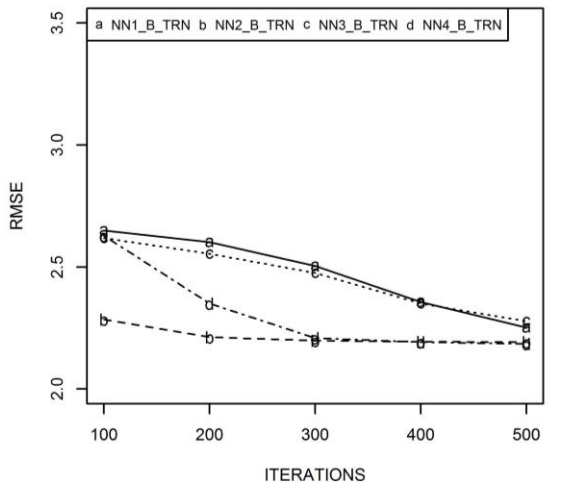

Figure 3: RMSE values obtained while training dataset B

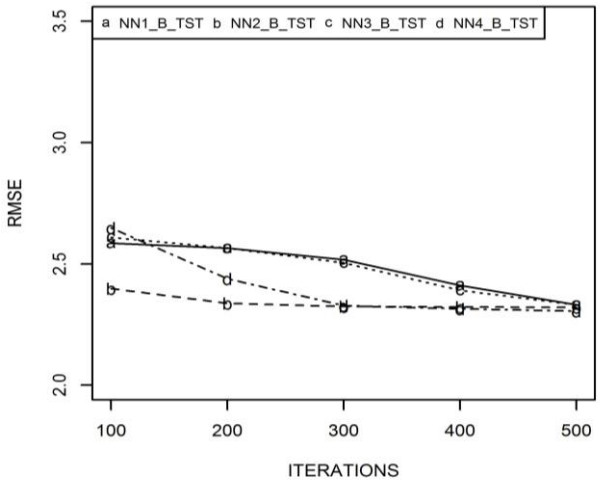

Figure 4: RMSE values obtained while testing dataset B

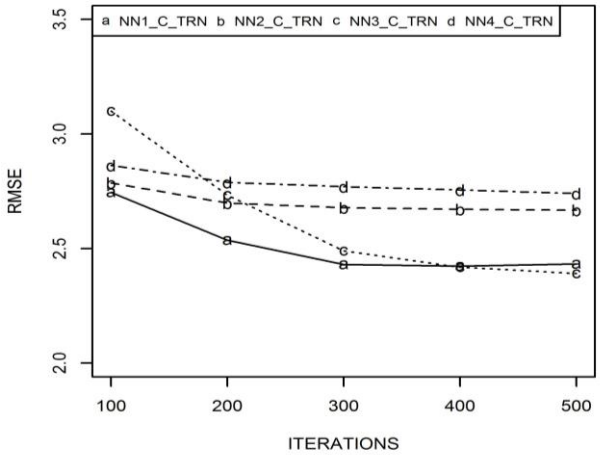

Figure 5: RMSE values obtained while training dataset C

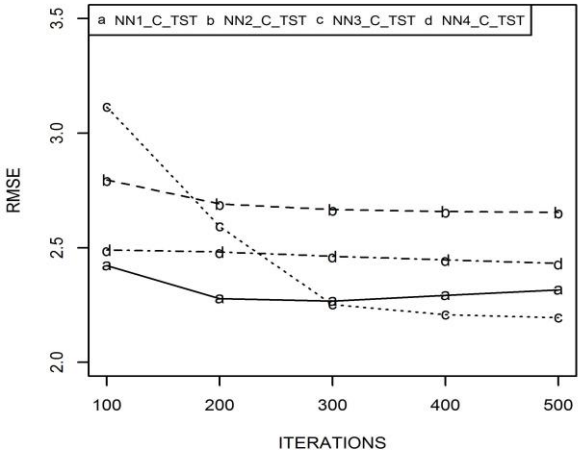

Figure 6: RMSE values obtained while testing dataset C

Table 2.1 : RMSE values obtained for neural networks for NN1_A, NN1_B and NN1_C

Neural Network	Iterations	RMSE Training	RMSE Testing
NN1-A	100	3.071138	2.24528
	200	3.023078	2.151095
	300	2.935038	1.694586
	400	2.788507	1.505884
	500	2.666433	1.679697
NN1_B	100	2.649884	2.584740
	200	2.601444	2.565257
	300	2.504793	2.517000
	400	2.355918	2.411118
	500	2.252201	2.331834
NN1_C	100	2.745243	2.422055
	200	2.536556	2.277851
	300	2.430734	2.266930
	400	2.423671	2.292304
	500	2.433085	2.316286

Table 2.2 : RMSE values obtained for neural networks for NN2_A, NN2_B and NN2_C

Neural Network	Iterations	RMSE Training	RMSE Testing
NN2_A	100	3.173132	1.291783
	200	2.884746	1.504091
	300	2.780341	1.628322
	400	2.735443	1.689284
	500	2.713498	1.718103
NN2_B	100	2.285066	2.398728
	200	2.212741	2.337107
	300	2.197964	2.324483
	400	2.193799	2.321519
	500	2.192254	2.320952
NN2_C	100	2.786705	2.796688
	200	2.698624	2.691312
	300	2.677617	2.665671
	400	2.670759	2.657319
	500	2.667830	2.653899

Table 2.3 : RMSE values obtained for neural networks for NN3_A, NN3_B and NN3_C

Neural Network	Iterations	RMSE Training	RMSE Testing
NN3-A	100	4.451919	1.686446
	200	3.724761	1.300172
	300	3.021242	1.596912
	400	2.798949	1.670685
	500	2.691488	1.785912
NN3_B	100	2.618055	2.608513
	200	2.554602	2.565001
	300	2.476058	2.504422
	400	2.350214	2.391568
	500	2.277369	2.331631
NN3_C	100	3.099950	3.114563
	200	2.733060	2.592882
	300	2.490171	2.251587
	400	2.418034	2.207532
	500	2.390984	2.194133

Table 2.4: RMSE values obtained for neural networks for NN4_A, NN4_B and NN4_C

Neural Network	Iterations	RMSE Training	RMSE Testing
NN4-A	100	3.759617	1.400096
	200	3.383205	1.389424
	300	3.042155	1.533281
	400	2.98491	1.567732
	500	2.975123	1.568552
NN4_B	100	2.628391	2.648764
	200	2.350734	2.4391
	300	2.20919	2.327273
	400	2.191522	2.314793
	500	2.184389	2.304893
NN4_C	100	2.862811	2.490486
	200	2.788462	2.481677
	300	2.769823	2.462799
	400	2.755091	2.447117
	500	2.739517	2.43169

VII RESULTS AND DISCUSSION

The Tables 2.1,2.2,2.3 and 2.4 shows the neural networks used, number of iterations, and RMSE values for training and testing. Figure 1, shows the graph plotted with number of iterations and the RMSE values for training of the networks NN1_A,NN2_A, NN3_A and NN4_A. Figure 2, shows the graph plotted with number of iterations and the RMSE values for testing of the networks NN1_A,NN2_A, NN3_A and NN4_A. Figure 3, shows the graph plotted with number of iterations and the RMSE values for training of the networks NN1_B,NN2_B, NN3_B and NN4_B. Figure 4, shows the graph plotted with number of iterations and the RMSE values for testing of the networks NN1_B,NN2_B, NN3_B and NN4_B. Figure 5, shows the graph plotted with number of iterations and the RMSE values for training of the networks NN1_C, NN2_C, NN3_C and NN4_C. Figure 6, shows the graph plotted with number of iterations and the RMSE values for testing of the networks NN1_C,NN2_C, NN3_C and NN4_C.

For the neural network NN1_C, we took initial values for w_i and b as zero. For five hundred iterations, the values of w_is are taking values as w_1 is 0.9944817, w_2 is 1.408236, w_3 is 1.241141 , w_4 is1.493772 , w_5 is -3.943402, w_6 is -1.319025, w_7 is 3.838257, b is -1.356216 and a is 18.03083. For the neural network NN2_C, again we took initial values for w_i and b as zero. For five hundred iterations, the values of w_is are taking values as w_1 is 1.295452, w_2 is 1.225783, w_3 is 1.177189 , w_4 is 1.129963, w_5 is -2.504797, w_6 is -1.663339 , w_7 is 2.576157 and b is -1.547379.

The difference between neural networks NN1 and NN2 is that the parameter *a* is changed during training in NN1 and the parameter *M* is fixed in NN2. It is observable that the effect of fixing at a value of a parameter is thrusted upon other parameters to arrive at the reduced error. Since other parameters and functions are same except *a* and *M*, the effects of it can be seen in the values obtained by other parameters w_i and b obtained by the neural networks NN1_C and NN2_C.

From the charts, it is observable that for large size of the dataset taken, an artificial neural network NN3 introduced in this research work with a different activation function is showing less root mean square error value over the other architectures both for training and testing. It is evident that for data set C, for five hundred iterations, for training the neural networks, NN3 followed by neural network NN1 has shown good results over other two networks. In the same data set and iterations, for testing again the same neural networks, NN3 followed by NN1 has shown good results. It is also observed from the graph that for data set A, for five hundred iterations, and for training of the network the NN1_A followed by NN3_A has performed well. In the same data set and for same number of iterations, for testing the NN4_A followed by NN1_A has shown lowest RMSE values. For data set B, for training NN4_B followed by NN2_B are showing lowest RMSE values. Again, the neural networks NN4 followed byNN2 are showing lowest RMSE values for testing in data set B.

CONCLUSION

We used four different neural networks to predict the attribute value from other attribute values in the given data set. We used three different neural networks using three different activation functions with no hidden layers and one neural network using one hidden layer in which hidden layer and output layer used two different activation functions. By changing the activation functions, we are able to find the changes in RMSE values. Over all it can be observed that for the large size of the data set taken for the experiment, that is for data set C, neural network introduced in this research NN3 has performed well both for training and testing of the dataset, followed by NN1, at least, as far as the data set taken and the networks formulated here are concerned. There are many other activation functions are available. The neural network architectures experimented here may be used to experiment other dataset and find whether there is any significant increase in the results produced with other known neural networks architectures. We can also use other activation functions on this data set and others to find whether there is any improvement in the results produced.

REFERENCES

[1] Zhang, G.P., 2000. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *30*(4), pp.451-462.

[2] Ki-Young Lee, Kyu-Ho Kim, Jeong-Jin Kang, Sung-Jai Choi, Yong-Soon Im, Young-Dae Lee, Yun-Sik Lim, Comparison and Analysis of Linear Regression & Artificial Neural Network, International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 20 (2017) pp. 9820-9825.

[3] Nayak, D.R., Mahapatra, A. and Mishra, P., 2013. A survey on rainfall prediction using artificial neural network. *International Journal of Computer Applications*, *72*(16).

[4] Ince, T., Kiranyaz, S., Pulkkinen, J. and Gabbouj, M., 2010. Evaluation of global and local training techniques over feed-forward neural network architecture spaces for computer-aided medical diagnosis. *Expert Systems with Applications*, *37*(12), pp.8450-8461.

[5] Liu, Y., Yang, J., Li, L. and Wu, W., 2012. Negative effects of sufficiently small initialweights on back-propagation neural networks. *Journal of Zhejiang University SCIENCE C*, *13*(8), pp.585-592.

[6] Rajan Dharwal and Loveneet Kaur, Applications of Artificial Neural Networks: A Review, Indian Journal of Science and Technology, Vol 9(47), December 2016.

[7] Zhang, J.R., Zhang, J., Lok, T.M. and Lyu, M.R., 2007. A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training. *Applied mathematics and computation*, *185*(2), pp.1026-1037.

[8] Hagan, M.T., Demuth, H.B., Beale, M.H. and De Jesús, O., 1996. *Neural network design* (Vol. 20). Boston: Pws Pub..

[9] Smetanin, Y.G., 1998. Neural networks as systems for recognizing patterns. *Journal of Mathematical Sciences*, *89*(4), pp.1406-1457.

[10] Zhang, G., Patuwo, B.E. and Hu, M.Y., 1998. Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, *14*(1), pp.35-62.

[11] Ding, W., Zhang, J. and Leung, Y., 2016. Prediction of air pollutant concentration based on sparse response back-propagation training feedforward neural networks. *Environmental Science and Pollution Research*, *23*(19), pp.19481-19494.

[12] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan and Stephen Marshall,Activation Functions: Comparison of trends in Practice and Research for Deep Learning,arXiv,2018, url=http://arxiv.org/abs/1811.03378.

[13] Lubna Shaikh and Kirti Sawlani, A rainfall prediction model using artificial neural network, International Journal of Technical Research and Applications Volume 5, Issue 2 (March - April 2017), PP. 45-48.

[14] Hung, N.Q., Babel, M.S., Weesakul, S. and Tripathi, N.K., 2009. An artificial neural network model for rainfall forecasting in Bangkok, Thailand. *Hydrology and Earth System Sciences*, *13*(8), pp.1413-1425.

[15] Cannady, J., 1998, October. Artificial neural networks for misuse detection. In *National information systems security conference* (Vol. 26).

[16] Varadraj P. Gurupur, Shrirang A. Kulkarni, Xinliang Liu, Usha Desai & Ayan Nasir (2018): Analysing the power of deep learning techniques over the traditional methods using medicare utilisation and provider data, Journal of Experimental & Theoretical Artificial Intelligence, DOI: 10.1080/0952813X.2018.1518999

[17] Chandra P, 2003, Sigmoidal Function Classes for Feedforward Artificial Neural Networks, Neural Processing Letters 18: 185–195, 2003.

[18] Njikam, A.N.S. and Zhao, H., 2016. A novel activation function for multilayer feed-forward neural networks. *Applied Intelligence*, *45*(1), pp.75-82

[19] Gomes, G.S.D.S., Ludermir, T.B. and Lima, L.M., 2011. Comparison of new activation functions in neural network for forecasting financial time series. *Neural Computing and Applications*, *20*(3), pp.417-439.

[20] Efe, M.Ö., 2008. Novel neuronal activation functions for feedforward neural networks. *Neural processing letters*, *28*(2), pp.63-79

[21] Suttisinthong, N., Seewirote, B., Ngaopitakkul, A. and Pothisarn, C., 2014, March. Selection of proper activation functions in back-propagation neural network algorithm for single-circuit transmission line. In *Proceedings of the International MultiConference of Engineers and Computer Scientists* (Vol. 2, pp. 10-14).

[22] Riedmiller, M., 1994. Advanced supervised learning in multi-layer perceptrons-from backpropagation to adaptive learning algorithms. Computer standards and interfaces, 16(3), pp.265-278.

[23] Asuncion, A. and Newman, D.J., 2007. UCI Machine Learning Repository [http://www.ics.uci. edu/~ mlearn/MLRepository.html]. Irvine, CA: University of California. *School of Information and Computer Science*, *12*.