Analysis of Artificial Neural Network for Financial Time Series Forecasting

Anupam Tarsauliya
Researcher
IIITM
Gwalior

Shoureya Kant
Researcher
IIITM
Gwalior

Rahul Kala
Researcher
IIITM
Gwalior

Ritu Tiwari
Asst. Preofessor
IIITM
Gwalior

Anupam Shukla
Professor
IIITM
Gwalior

ABSTRACT

Financial forecasting has been challenging problem due to its high non-linearity and high volatility. An Artificial Neural Network (ANN) can model flexible linear or non-linear relations- hip among variables. ANN can be configured to produce desired set of output based on set of given input. In this paper we attempt at analyzing the usefulness of artificial neural network for forecasting financial data series with use of different algorithms such as backpropagation, radial basis function etc. With their ability of adapting non-linear and chaotic patterns, ANN is the current technique being used which offers the ability of predicting financial data more accurately. "A x-y-1 network topology is adopted because of x input variables in which variable y was determined by the number of hidden neurons during network selection with single output." Both x and y were changed.

Keywords

ANN, Financial forecasting, BPA, LRN, RBF, GRNN

1. INTRODUCTION

Forecasting is a process that produces a set of outputs by a given set of input variables. The variables are normally historical data [1]. Basically, forecasting assumes that future occurrences are based, at least in part, on presently observable or past events. It assumes that some aspects of the past patterns will continue into the future. Past relationships can then be discovered through study and observation. The basic idea of forecasting is to find an approximation of mapping between the input and output data in order to discover the implicit rules governing the observed movements [2].

Stock market forecasting has always been a challenging problem. The source of its difficulty is the complex interactions between the market-influencing factors and the unknown random processes like unexpected news or other sudden changes in the influencing factors. On the other hand, there is some risk to investment in the stock market due to its unpredictable behaviors. Thus, an 'intelligent' prediction model for financial data forecasting would be deeply desired and would of wider interest. ANNs are relatively recent method for business forecasting [3]. The success of ANN applications can be qualified of their features and powerful pattern recognitions capability. The use of ANN in this field has been growing due to their ability to model complex nonlinear systems on sample data. An ANN is a new kind of computing tool that is not limited by equations or rules. ANN functions by finding correlations and patterns in the data which you provide. These patterns become a part of the network during training [4]. A separate network might be needed for each problem you want to solve, but many networks follow the same basic format.

Structure of the network affects the accuracy of the forecast. Network configuration mainly depends on the number of hid- den layers, number of neurons in each hidden layer, number of input neurons and the selection of activation function. No clear cut guide lines exist up to date for deciding the architecture of ANN. Mostly it is problem dependent. An ANN has to be con- figured such that the application of a set of inputs produces the desired set of outputs. Various methods to set the strengths of the connections exist. One way is to set the weights explicitly, using a priori knowledge. Another way is to train the neural network by feeding it teaching patterns and letting it change its weights according to some learning rule [5]. The learning situations in neural networks may be classified into three distinct sorts. In learning, an input vector is presented at the inputs together with a set of desired responses, one for each node, at the output layer. A forward pass is done, and the errors or discrepancies between the desired and actual response for each node in the output layer are found. These are then used to determine weight changes in the net according to the prevailing learning rule. These networks have self-learning capability and are fault-tolerant as well as noise-immune, and also have applications in various fields like system identification, pattern recognition, classification, speech recognition, image processing, etc.

Back propagation is a form of supervised learning for multi-layer nets, also known as the generalized delta rule. Error data at the output layer is "back propagated" to earlier ones, allowing incoming weights to these layers to be updated. It is most often used as training algorithm in current neural network applications. The back propagation algorithm was developed by Paul Werbos in 1974 and rediscovered independently by Rumelhart and Parker. Since its rediscovery, the back propagation algorithm has been widely used as a learning algorithm in feed forward multilayer neural networks. What makes this algorithm different than the others is the process by which weights are calculated during the learning network. In general, the difficulty with multilayer Perceptrons is calculating weights of the hidden layers in an efficient way that result in the least (or zero) output error; the more hidden layers there are, the more difficult it becomes. To update the weights, one must calculate an error. At the output layer this error is easily measured; this is the difference between the actual and desired outputs. At the hidden layers, however, there is no direct observation of the error; hence, some other technique must be used. To calculate an error at the hidden layers that will cause minimization of the output error, as this is the ultimate goal. The backpropagation algorithm is an involved mathematical tool; however, execution of the training equations is based on iterative processes, and thus is easily implementable on a computer.

Several studies relating to ANN and statistical models have been conducted in the literature. Traditional forecasting methods are limited in their effectiveness as they make assumptions about the distribution of the underlying data, and often fail to recognize the interrelatedness of variables [6]. Both linear and nonlinear models were used to predict stock returns [7] who emphasize the Nonlinear Model proving to be more effective. Such studies prove that the nonlinear model presents more consistent results for stock exchange market. For this reason, ANN applications have been widely used in a variety of areas in financial markets [8], [9]. Reference [9] confirmed that ANN was used for the solution of numerous financial problems. References [10], [11], [12] emphasized that ANN could be used in the prediction of financial markets, in particular, the prediction of stock market indexes which are considered to be a barometer of the markets in many countries. Empirical evidence suggests that although these models appear to be capable of explaining the movements of major exchange rates in the long run and in economies experiencing hyperinflation, their performance is poor when it comes to the short run and out-of-sample forecasting [13]. Conventional time series models forecasting on global approximation models, employing techniques such as linear and non-linear regression, polynom- ial fitting and artificial neural networks. Such models are better suited to problems with stationary dynamics [14]. In [15] and [16] the application of unsupervised clusters for the segmentation of the input space, and feed forward neural networks (FNNs) acting as local predictors for each identified cluster, was proposed. Neural network researchers and developers using the generalized method for determining the mini-mum necessary training set size will be able to implement neural networks with the highest forecasting performance at the least cost [17].

2. METHODOLOGY

We load the given time series dataset (un-normalized) into the system for its forecasting. For the loaded dataset, we bifurcate dataset into training and testing datasets respectively. A random dataset division is followed to result 70% of dataset as training dataset and remaining 30% as testing dataset. Training dataset is the outcome of random method followed to bifurcate the loaded dataset. Training dataset is used for defining the architecture of the neural network and train the defined neural network based on its data to predict the dataset. Testing dataset thus obtained is used for simulating the trained network, checking the error or accuracy of the trained network. We compare the output data as given by the network with the testing data set. The results of these comparisons are dealt in detail in later part of the paper.

The training dataset which is the 70% of the dataset is first converted into logarithmic form and then are normalized. The datasets are normalized using the general normalization formulae. The datasets are then fed into the network and are trained through various training algorithms which are described later in the paper. During the training phase the number of hidden neurons, the epochs, the momentum etc are altered and the network weights and biases are set as per these alterations. After the network has been trained we perform the testing phase on the new defined network. The remaining 30% of the dataset which is defined for the testing purpose is fed into the new trained network and is simulated accordingly. The block diagram of the methodology is shown in figure 1.

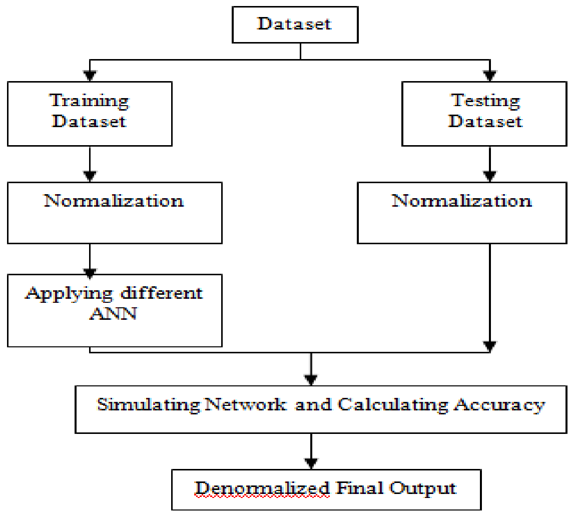

Figure1. Flow Chart for Methodology

The above mentioned data sets are taken and are processed through the methodology stated above. Each data set is exposed to different algorithms and is trained accordingly. The different algorithms used are as follows:

2.1 Back Propagation Algorithm (BPA)

Backpropagation is the generalization of the Widrow-Hoff learning rule to multiple-layer networks and nonlinear differentiable transfer functions [20]. Input vectors and the corresponding target vectors are used to train a network until it can approximate a function, associate input vectors with spec- ific output vectors, or classify input vectors in an appropriate way as defined by you. Networks with biases, a sigmoid layer, and a linear output layer are capable of approximating any function with a finite number of discontinuities.

Standard backpropagation is a gradient descent algorithm, as is the Widrow-Hoff learning rule, in which the network weights are moved along the negative of the gradient of the performance function. The term backpropagation refers to the manner in which the gradient is computed for nonlinear multilayer networks. There are a number of variations on the basic algorithm that are based on other standard optimization techniques, such as conjugate gradient and Newton methods.

2.2 Layer Recurrent Network (LRN):

In the LRN, there is a feedback loop, with a single delay, around each layer of the network except for the last layer. The original Elman network had only two layers, and used a tansig transfer function for the hidden layer and a purelin transfer function for the output layer. The original Elman network was trained using an approximation to the backpropagation algorithm. The newlrn command generalizes the Elman network to have an arbitrary number of layers and to have arbitrary transfer functions in each layer.

2.3 Radial basis network (RBN)

The function newrb iteratively creates a radial basis network one neuron at a time. Neurons are added to the network until the sum-squared error falls beneath an error goal or a maximum number of neurons has been reached. The function newrb takes matrices of input and target vectors P and T, and design parameters goal and spread, and returns the desired network.

2.4 Generalized Regression Networks (GRNN)

A generalized regression neural network is often used for function approximation. It has a radial basis layer and a spec- ial linear layer. The output of the network is in the normalized form, so we first denormalize the output and then its exponent is taken. After the following process is done the output comes in a form comparable to the original dataset. Now the comparison of the input and the output dataset is done and the results are explained in details in the later part of the paper.

3. RESULTS

3.1 Research Data

We have used two different data sets for our research. The data (un-normalized) have been collected from Prof. Rob J Hyndman’s website http://robjhyndman.com/TSDL/. Data sets analyzed are as: Daily closing price of IBM stock, Jan. 01 1980 - Oct. 08 1992. Source: [18], Daily S & P 500 index of stocks, Jan. 01 1980 - Oct. 08 1992. Source: [18]. Table 1 summarizes the stated information.

The first few data indexes of series are used for the research. For training, 70% of the data of the series has been used and remaining 30% is used for testing.

Table 1. Time Series Data Sets

Time Series	Standard Deviation	Mean	Count
Daily IBM	5.736916	60.89908	500
Daily S&P	10.1308	123.3728	500

3.2 Data Analysis

3.2.1 Backpropagation Algorithm (BPA)

Below is the table for marking efficient artificial neural network architecture for different data series with backpropagation as training algorithm. We have set the input vector of constant neurons with specified value of learning rate and momentum. Epochs are kept constant at 3000. We started optimizing architecture by gradually increasing the number of hidden neurons. As the number of hidden neurons increase the mean square error first decreases gradually and then starts increasing. The variation of the network output is depicted in the table. The value of the hidden neuron for which the most optimum result is obtained and is taken into consideration for further optimizing it with learning rate and momentum.

After the obtained optimized values for hidden neurons, the learning rate and momentum are optimized. These values are used for obtaining the optimum input vector set. The input vector is gradually increased. The mean square error simultaneously increases and then decreases. The most optimum neural network architecture for backpropagation training algorithm is obtained for the two datasets. The analysis for the two datasets is given below.

Table 2 and Table 3 show the analysis for daily IBM.

Table 2. Diff. ANN Arch. for Daily IBM using BPA at epochs=3000

Different ANN Architecture				
x-y-1	**lr**	**mc**	**Mean**	**S.D.**
10-2-1	0.5	0.7	2.79652	0.54519
10-5-1	0.5	0.7	2.08174	0.346772
10-10-1	0.5	0.7	2.17914	0.093233
10-20-1	0.5	0.7	2.17464	0.297468
10-30-1	0.5	0.7	2.56222	0.246952
10-5-1	0.3	0.7	1.98028	0.463858
10-5-1	0.7	0.7	1.95362	0.487149

Table 3. Diff. ANN Arch. varying inputs for Daily IBM using BPA at epochs=3000

Different Number Of Inputs				
x-y-1	**lr**	**mc**	**Mean**	**S.D.**
05-5-1	0.3	0.7	2.08604	0.378809
08-5-1	0.3	0.7	2.20682	0.386111
10-5-1	0.3	0.7	1.98028	0.463858
15-5-1	0.3	0.7	2.68886	0.433932
20-5-1	0.3	0.7	2.4397	0.332563

Table 4 shows the most optimal structure for daily IBM data series.

Table 4. Optimum table

x-y-1	**lr**	**mc**	**Mean**	**S.D.**
10-05-1	0.3	0.7	1.98028	0.463858

Table 5 and Table 6 show the analysis for daily S&P.

Table 5. Diff. ANN Arch. for Daily S&P using BPA at epochs=3000

Different NN Architecture				
x-y-1	**lr**	**mc**	**Mean**	**S.D.**
10-2-1	0.5	0.7	6.28158	2.078473
10-5-1	0.5	0.7	3.6688	0.745425
10-10-1	0.5	0.7	3.52292	0.555277
10-20-1	0.5	0.7	3.43212	0.792911
10-30-1	0.5	0.7	5.19912	0.861521
10-20-1	0.3	0.7	4.59012	0.697043
10-20-1	0.7	0.7	3.73926	0.768024
10-20-1	0.8	0.7	3.15062	0.592651
10-5-1	0.8	0.5	3.5666	0.692245
10-5-1	0.8	0.9	3.91194	0.80667

Table 6. Diff. ANN Arch. varying inputs for Daily S&P using BPA at epochs=3000

Different Number Of Inputs				
x-y-1	**lr**	**mc**	**Mean**	**S.D.**
05-20-1	0.8	0.7	3.51574	1.119034
08-20-1	0.8	0.7	3.03788	0.629401
10-20-1	0.8	0.7	3.15062	0.592651
15-20-1	0.8	0.7	3.78	0.944653
20-20-1	0.8	0.7	4.06736	0.833488

Table 7 shows the most optimal structure for daily S&P data series.

Table 7. Optimum Table

x-y-1	lr	mc	Mean	S.D.
8-20-1	0.8	0.7	3.03788	0.629401

3.2.2 Layer-Recurrent Network (LRN)

Below are the tables for marking efficient artificial neural network architecture for different data series with Layer-Recurrent Network as training algorithm. We have set the input vector of constant neurons with specified value. Epochs are kept constant at 30. We started optimizing architecture by gradually increasing the number of hidden neurons. As the number of hidden neurons increase the mean square error first decreases gradually and then starts increasing and again decreases. It shoes abrupt changes in the rmse. The variation of the network output is depicted in the table. The value of the hidden neuron for which the most optimum result is obtained and is taken into consideration for further optimizing it with learning rate and momentum.

After the obtained optimized values for hidden neurons, the input vector set is optimized. The input vector is gradually increased. The mean square error decreases and then increases. The most optimum neural network architecture for Layer-Recurrent Network training algorithm is obtained for the two datasets. The analysis tables for the two datasets are given below.

Table 8 and Table 9 show the analysis for daily IBM.

Table 8. Diff. ANN Arch. varying hidden neurons for Daily IBM using LRN

Different NN architecture		
x-y-1	**RMSE**	**S.D.**
10-1-1	0.96118	0.10258995
10-3-1	0.92746	0.08610867
10-5-1	2.13334	2.02196345
10-8-1	1.93962	2.18422513
10-10-1	0.9874	0.06050483
10-15-1	2.74854	2.54473861

Table 9. Diff. ANN Arch. varying inputs for Daily IBM using LRN

Different Number Of Inputs		
x-y-1	**RMSE**	**S.D.**
5-3-1	1.04054	0.07314635
8-3-1	0.90856	0.09027615
10-3-1	0.92746	0.08610867
15-3-1	2.91092	2.73857886
20-3-1	5.52258	0.63822251

Table 10 shows the most optimal structure for daily IBM data series.

Table 10. Optimum Table

x-y-1	RMSE	S.D.
8-3-1	0.90856	0.09027615

Table 11 and Table 12 show the analysis for daily S&P.

Table 11. Diff. ANN Arch. varying hidden neurons for Daily S&P using LRN

Different NN architecture		
x-y-1	**RMSE**	**S.D.**
10-1-1	1.13512	0.08749358
10-3-1	1.15496	0.17080976
10-5-1	1.05572	0.05598077
10-8-1	1.18746	0.04093053
10-10-1	1.05518	0.09418026
10-15-1	1.0931	0.12033027

Table 12. Diff. ANN Arch. varying inputs for Daily S&P using LRN

Different input		
x-y-1	**RMSE**	**S.D.**
5-10-1	1.19688	0.11266555
8-10-1	1.1948	0.17505239
10-10-1	1.05518	0.09418026
15-10-1	1.20526	0.19910125
20-10-1	1.1028	0.07104798

Table 13 shows the most optimal structure for daily S&P data series.

Table 13. Optimum Table

x-y-1	RMSE	S.D.
10-10-1	1.05518	0.09418026

3.2.3 Radial Basis Network (RBN):

Below are the tables for marking efficient artificial neural network architecture for different data series with Radial basis network as training algorithm. We have set the input vector of constant neurons with specified value. We started optimizing architecture by gradually increasing the spread. As the value of spread increase the mean square error first increases gradually and then starts decreasing. The variation of the network output is depicted in the table. The value of the spread for which the most optimum result is obtained and is taken into consideration for further optimizing it with learning rate and momentum.

After the obtained optimized values of spread, the input vector set is optimized. The input vector is gradually increased. The mean square error simultaneously increases and then decreases. The most optimum neural network architecture for Radial basis network training algorithm is obtained for the two datasets. The analysis tables for the two datasets are given below.

Table 14 and Table 15 show the analysis for daily IBM.

Table 14. Diff. ANN Arch. varying spread for Daily IBM using RBN

Different NN Architecture			
x-y-1	**Spread**	**RMSE**	**S.D.**
10-y-1	2	1.18628	0.203878
10-y-1	5	1.20408	0.234053
10-y-1	8	1.10316	0.084471
10-y-1	10	1.18188	0.176929
10-y-1	15	1.02964	0.079143

Table 15. Diff. ANN Arch. varying inputs for Daily IBM using RBN

Different Input			
x-y-1	**Spread**	**RMSE**	**S.D.**
5-y-1	15	1.15466	0.104459
8-y-1	15	1.30836	0.481532
10-y-1	15	1.02964	0.0791143
15-y-1	15	1.13186	0.147249
20-y-1	15	1.23578	0.256903

Table 16 shows the most optimal structure for daily IBM data series.

Table 16. Optimum Table

x-y-1	Spread	RMSE	S.D.
10-y-1	15	1.02964	0.0791143

Table 17 and Table 18 show the analysis for daily S&P.

Table 17. Diff. ANN Arch. varying spread for Daily S&P using RBN

Different NN Architecture			
x-y-1	**Spread**	**RMSE**	**S.D.**
10-y-1	2	1.66626	0.23091078
10-y-1	5	1.5653	0.25426902
10-y-1	8	2.18502	0.96924005
10-y-1	10	1.79984	0.87878396
10-y-1	15	1.99958	0.67472272

Table 18. Diff. ANN Arch. varying inputs for Daily S&P using RBN

Different input			
x-y-1	**Spread**	**RMSE**	**S.D.**
5-y-1	5	1.68234	0.23684954
8-y-1	5	1.9791	0.56714218
10-y-1	5	1.5653	0.25426902
15-y-1	5	1.81382	0.16263172
20-y-1	5	1.76686	0.19976665

Table 19 shows the most optimal structure for daily S&P data series.

Table 19. Optimum Table

x-y-1	Spread	RMSE	S.D.
10-y-1	5	1.5653	0.25426902

3.2.4 Generalized Regression Networks (GRNN):

Below are the tables for marking efficient artificial neural network architecture for different data series with Generalized Regression Networks as training algorithm. The input vector is gradually increased due to which the rmse decreases. The most optimized input set is taken into consideration for the network architecture. The analysis tables for the two datasets are given below.
Table 20 shows the analysis for daily IBM.

Table 20. Diff. ANN Arch. for Daily IBM using GRNN

Different NN architecture		
x-y-1	**RMSE**	**S.D.**
5-10-1	5.87704	0.13995588
8-16-1	5.85182	0.19353296
10-20-1	5.8453	0.29619885
15-30-1	5.62394	0.1269295
20-40-1	5.5722	0.10498417

Table 21 shows the most optimal structure for daily IBM data series.

Table 21. Optimum Table

x-y-1	RMSE	S.D.
20-40-1	5.5722	0.10498417

Table 22 shows the analysis for daily S&P.

Table 22. Diff. ANN Arch. for Daily S&P using GRNN

Different NN architecture		
x-y-1	**RMSE**	**S.D.**
5-10-1	9.98948	0.30826209
8-16-1	10.10648	0.30243149
10-20-1	9.72556	0.26012198
15-30-1	9.55536	0.48860856
20-40-1	10.3707	0.89469728

Table 23 shows the most optimal structure for daily S&P data series.

Table 23. Optimum Table

x-y-1	RMSE	S.D.
15-30-1	9.55536	0.48860856

3.3 Comparison

3.3.1 Daily IBM

The most optimum ANN architecture and input parameter for different types of ANN used is analyzed. Table XXIV shows the comparative analysis of different ANN with respect to the Daily IBM time series. Table 24 shows comparative analysis for daily IBM.

Table 24. Comparison Table for Daily IBM

Method	Architecture	Mean	Standard Deviation
BPA	10-5-1	1.98028	0.463858
LRN	8-3-1	0.90856	0.09027615
RBN	10-y-1	1.02964	0.0791143
GRNN	20-40-1	5.5722	0.10498417

3.3.2 Daily S&P

The most optimum ANN architecture and input parameter for different types of ANN used is analyzed. Table XXV shows the comparative analysis of different ANN with respect to the Daily IBM time series. Table 25 shows comparative analysis for daily S&P.

Table 25. Comparison Table for Daily S&P

Method	Architecture	Mean	Standard Deviation
BPA	8- 20- 1	3.03788	0.629401

LRN	10-10-1	1.05518	0.09418026
RBN	10-y-1	1.5653	0.25426902
GRNN	15-30-1	9.55536	0.48860856

4. GRAPHICAL ANALYSIS

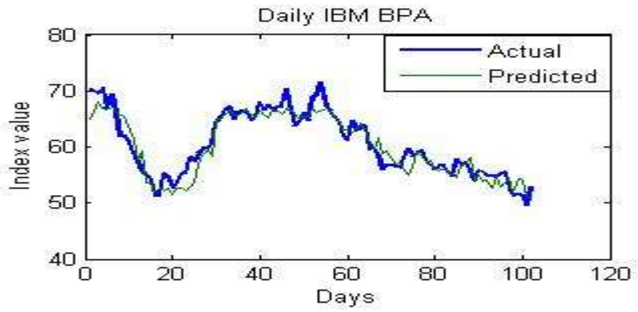

Figure 2. Graph for actual and predicted values for Daily IBM using BPA

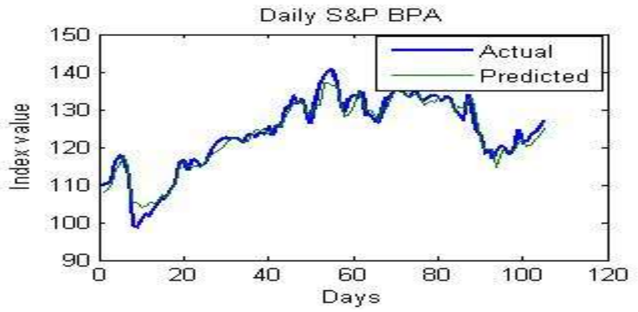

Figure 3. Graph for actual and predicted values for Daily S&P using BPA

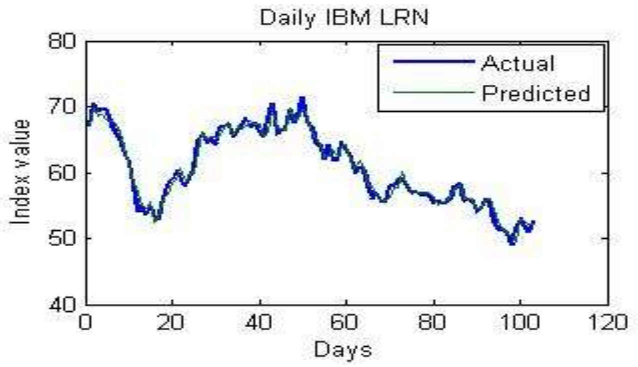

Figure 4. Graph for actual and predicted values for Daily IBM using LRN

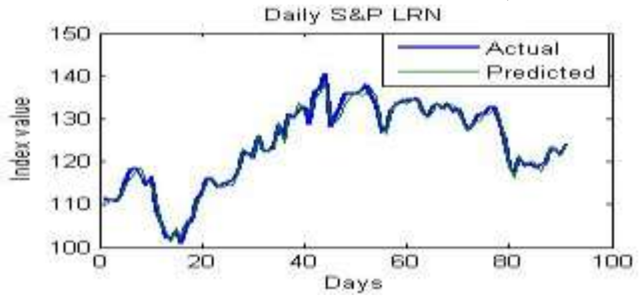

Figure 5. Graph for actual and predicted values for Daily S&P using LRN

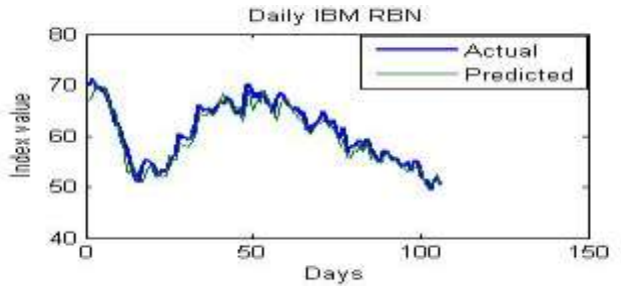

Figure 6. Graph for actual and predicted values for Daily IBM using RBN

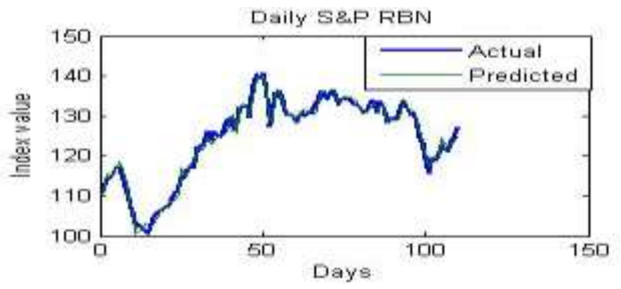

Figure 7. Graph for actual and predicted values for Daily S&P using RBN

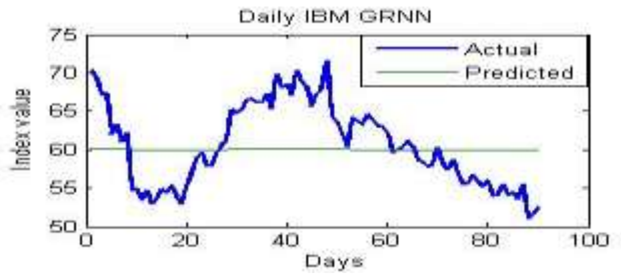

Figure 8. Graph for actual and predicted values for Daily IBM using GRNN

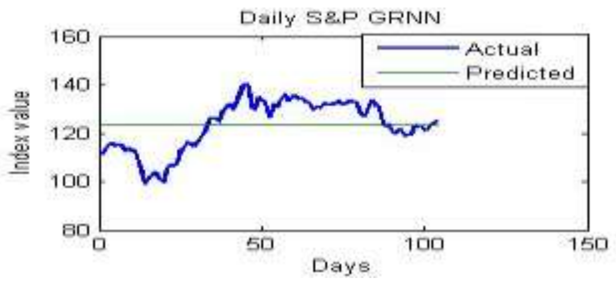

Figure 9. Graph for actual and predicted values for Daily S&P using GRNN

5. CONCLUSIONS

This paper attempts at analyzing the usefulness of artificial neural network for forecasting financial data series with use of different algorithms such as backpropagation, radial basis function etc. A x-y-1 network topology is adopted because of x input variables in which variable y was determined by the number of hidden neurons during network selection with single output." Both x and y were changed. Following conclusions could be drawn from the empirical results and comparison graph plotted between actual and predicted index value.

- Time series prediction probability over all datasets can be analyzed reasonably by number of neurons as compared to other problems.
- Increasing the number of hidden neurons first decreases rmse and then increases it.
- Increasing number of input neurons first decreases and then increases the rmse.
- Results may be generalizable to all the data sets.

6. REFERENCES

[1] Zhou Yixin, Jie Zhang, "Stock Data Analysis Based on BP Neural Network," iccsn, pp.396-399, 2010 Second International Conference on Communication Software and Networks, 2010

[2] Marzi, H.; Turnbull, M.; Marzi, E.; , "Use of neural networks in forecasting financial market," Soft Computing in Industrial Applications, 2008. SMCia '08. IEEE Conference on , vol., no., pp.240-245, 25-27 June 2008

[3] Ming Hao Eng; Yang Li; Qing-Guo Wang; Tong Heng Lee; , "Forecast Forex with ANN Using Fundamental Data," Information Management, Innovation Management and Industrial Engineering, 2008. ICIII '08. International Conference on , vol.1, no., pp.279-282, 19-21 Dec. 2008

[4] Fangwen Zhai; Qinghua Wen; Zehong Yang; Yixu Song; , "Hybrid forecasting model research on stock data mining," New Trends in Information Science and Service Science (NISS), 2010 4th International Conference on , vol., no., pp.630-633, 11-13 May 2010

[5] Kar, B.; Mandal, K.K.; Pal, D.; Chakraborty, N.; , "Combined economic and emission dispatch by ANN with backprop algorithm using variant learning rate & momentum coefficient,"Power Engineering Conference, 2005. IPEC 2005. The 7th International , vol., no., pp.1-235, Nov. 29 2005-Dec. 2 2005

[6] Joarder Kamruzzaman, Ruhul A. Sarker, “ANN Based Forecasting of Foreign Currency Exchange Rates”, Neural Information Processing - Letters and Reviews Vol.3, No. 2, May 2004, pp/49

[7] A. Kanas, “Non-linear forecasts of stock returns ,” Journal of Forecasting, vol. 22, no.4, pp. 299–315, July 2003.

[8] R.S. Ludwig and M.J. Piovoso, “A Comparison of Machine-Learning Classifiers For Selecting Money Managers,” Intelligent Systems in Accounting, Finance and Management, Chichester, vol. 13, no. 3, p. 151-164, July 2005.

[9] K. Kumar and S. Bhattacharya, “Artificial Neural Network vs Linear Discriminant Analysis in Credit Ratings Forecast: A Comparative Study of Prediction Performances,” Review of Accounting and Finance, vol. 5, no. 3, 217-227, 2006.

[10] S.V. Kartalopoulos, Understanding Neural Networks and Fuzzy Logic: Basic Concepts and Applications, Wiley IEEE Press, New York, August 1995.

[11] R.Stein and V. Dhar, Intelligent Decision Support Methods: The Science of Knowledge Work, Prentice Hall Business Publishing, N.J., 1996

[12] S. Ward and M. Sherald, The Neural Network Financial Wizards, Technical Analysis of Stocks and Commodities, Reprinted, Technical Analyses Inc., Seattle, Washington 1995.

[13] J. A. Frankel and A. K. Rose, Empirical research on nominal exchange rates, Handbook of International Economics (G. Grossman and K. Rogo®, eds.), vol. 3, Amsterdam, North-Holland, 1995, pp. 1689-1729.

[14] A. S.Weigend, M. Mangeas, and A. N. Srivastava, Nonlinear gated experts for time series: Discovering regimes and avoiding over¯tting, International Journal of Neural Systems 6 (1995), 373-399.

[15] N. G. Pavlidis, D. K. Tasoulis, and M. N. Vrahatis, Financial forecasting through unsupervised clustering and evolutionary trained neural networks, Proceedings of the Congress on Evolutionary Computation (CEC 2003), 2003, pp. 2314-2321.

[16] N.G. Pavlidis, D.K. Tasoulis, V.P. Plagianakos, and M.N. Vrahatis, Computational intelligence methods for financial time series modeling, International Journal of Biffurcation and Chaos (accepted for publication) (2005).

[17] STEVEN WALCZAK, An Empirical Analysis of Data Requirements for Financial Forecasting with Neural Networks.

[18] Hipel and McLeod Time Series Modelling of Water Resources and Environmental Systems, 1994, Elsevier.

[19] Makridakis, Wheelwright and HyndmanForecasting: Methods and Applications, 3rd ed, 1998, Wiley Nelson.

[20] A. Shukla, R. Tiwari, R. Kala Real Life Application of Soft Computing, CRC Press 2010.